LLNL-TR-613612

LAWRENCE LIVERMORE NATIONAL LABORATORY

Adaptive AMG with Coarsening Based on Compatible Weighted Matching

P. D'Ambra, P. S. Vassilevski

January 28, 2013

This work performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

Computing and Visualization in Science manuscript No.
(will be inserted by the editor)

Adaptive AMG with coarsening based on compatible weighted matching

Pasqua D'Ambra · Panayot S. Vassilevski

Received: date / Accepted: date

Abstract We introduce a new composite adaptive Algebraic Multigrid (composite αAMG) method to solve systems of linear equations without a-priori knowledge or assumption on characteristics of near-null components of the AMG preconditioned problem referred to as *algebraic smoothness*. Our version of αAMG is a composite solver built through a bootstrap strategy aimed to obtain a desired convergence rate. The coarsening process employed to build each new solver component relies on a pairwise aggregation scheme based on maximum weighted matching in a graph and on principles of compatible relaxation. The latter replaces the commonly used characterization of strength of connection in both the coarse space selection and in the interpolation scheme. The goal is to design a method leading to scalable AMG for a wide class of problems that go beyond the standard elliptic Partial Differential Equations (PDEs). In the present work, we introduce the method and demonstrate its potential when applied to symmetric positive definite linear systems arising from finite element discretization of highly anisotropic elliptic PDEs on structured and unstructured meshes.

Keywords Adaptive AMG · weighted matching · strength of connection · compatible relaxation

This work of the second author was performed under the auspices of the U.S. Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344.

P. D'Ambra
Institute for High-Performance Computing and Networking, National Research Council of Italy, Naples, Italy.
E-mail: pasqua.dambra@cnr.it

P. S. Vassilevski
Center for Applied Scientific Computing, Lawrence Livermore National Laboratory, Livermore, CA, USA.
E-mail: panayot@llnl.gov

Mathematics Subject Classification (2000) 65F10, 65N55

1 Introduction

We are interested in solving large and sparse linear systems of equations

$$A\mathbf{x} = \mathbf{b},$$

where $A \in \mathbb{R}^{n\times n}$ is assumed symmetric positive definite (s.p.d.), by algebraic multigrid (AMG) and more specifically by aggregation based AMG. The AMG methods, originated in [5], together with the smoothed aggregation AMG (or SA AMG) [20], have become a powerful tool for solving problems of linear algebraic equations that typically arise from discretization of elliptic PDEs. In recent years substantial progress has been made to extend the applicability of AMG to more general sparse linear systems by developing methods that use appropriate adaptive strategies (cf., [2,3,7,8,14], etc.) that are aimed at capturing the near-null components of the error (sometimes referred to as *algebraically smooth* components) that the current solver cannot efficiently handle so that they are then used to improve the solver by modifying its hierarchy of coarse spaces.

The approach that we utilize builds upon the adaptive AMG ideas however presents several new features. It is also fairly general in the sense that we do not assume any specific knowledge of the near-nullspace of A (or of a preconditioned version of A, such as $B^{-1}A$). The main philosophy is the same as in the original adaptive AMG papers (cited above); namely we test the current method (represented by an operator B) applied to the trivial system $A\mathbf{x} = 0$ starting with a nonzero random initial iterate $\mathbf{x}$, by computing $\mathbf{x} :=$

$(I - B^{-1}A)\mathbf{x}$, which effectively provides an approximation to the eigenvector of $B^{-1}A$ corresponding to the minimal eigenvalue of $B^{-1}A$. If during this process a slow convergence is encountered, we use the most recent iterate to form a new coarse hierarchy. This is the first main difference with the previously studied adaptive AMG methods. As a result, we end up with a composite AMG solver B, given by the product formula

$$I - B^{-1}A = \prod_{j}(I - B_j^{-1}A),$$

where each B_j corresponds to a separate hierarchy constructed driven by a particular algebraically smooth vector.

Another difference in our approach is in the coarsening process employed to obtain a multilevel hierarchy. We consider coarsening by pairwise aggregation based on a maximum weighted matching (for definitions, see Section 2) applied to the matrix adjacency graph. At each level of the hierarchy, starting from a maximum product matching of the graph associated with the current matrix, we generate two complementary coarser vector spaces by simple piecewise constant interpolation of a given algebraically smooth vector. We select the coarse space based on the principles of compatible relaxation (originated in [1]), i.e., we test the convergence of a pointwise smoother on homogeneous systems associated to the two available coarser matrices and choose as new coarse matrix and new algebraically smooth vector those for which slower convergence is observed. In fact, if we use matching so that the aggregates gather together pairs of fine degrees of freedom (or dofs) that are "strongly connected" the complementary space gives rise to a hierarchically complement matrix that is well-conditioned (when preconditioned by the smoother). In general, the procedure can end up building a binary tree of multiple coarse spaces by matching-based aggregation where, at each level, selection of coarsening branch is based on compatible relaxation of a given vector. We use both optimal solution for maximum product matching and an approximation algorithm and demonstrate the performance of our adaptive AMG on the difficult (for multigrid) s.p.d. linear systems arising from discretization of anisotropic PDEs on structured and unstructured meshes. In particular, we demonstrate that our coarsening strategy clearly detects the direction of anisotropy in both structured and unstructured mesh cases.

The remainder of the paper is organized as follows. In Section 2, we recall the notion of graph associated to a sparse matrix and remind the relation between maximum product matching and linear algebra applications. Then we describe the algorithm for pairwise aggregation based on maximum product matching. In Section 3, we introduce two algebraic coarsening processes based on the pairwise aggregation, depending on the weights we used for matching. The actual coarse vector space is chosen based on compatible relaxation principles. In Section 4, we outline the bootstrap strategy employed to build a composite αAMG with a prescribed convergence rate, whereas in Section 5 we present an extensive set of numerical results illustrating our approach. Finally, some remarks and future work are included in Section 6.

2 Pairwise aggregation based on weighted matching

To find matching in a graph is a classical problem in combinatorial optimization which has wide range of applications in Sparse Linear Algebra [10]. The starting point is the representation of the sparse matrices in terms of graphs [18]. Let $A = (a_{ij})_{i,j=1,\ldots,n}$ be a sparse matrix, the *graph* associated with A is the pair $G_A = (V, E)$, where the vertex set V corresponds to the row/column indices of A and the edge set E corresponds to the set of nonzeros of the matrix A so that $(i, j) \in E$ iff $a_{ij} \neq 0$. In the case of matrices with symmetric sparsity pattern, the edges (i, j) are undirected pairs of vertices, i.e. $(i, j) = (j, i) \in E$ iff $a_{ij} \neq 0$ and $a_{ji} \neq 0$, and G_A is called *undirected graph*. In the case of a graph $G_B = \{V_r \cup V_c, E\}$, where the vertex set consists of two disjoint sets corresponding to the rows and columns of A, respectively, and $(i, j) \in E$ iff $a_{ij} \neq 0$ for $i \in V_r$ and $j \in V_c$, the graph is called *bipartite*. A *matching* $\mathcal{M} \subseteq E$ in a graph (G_A or G_B) is a set of edges such that no two edges share the same vertex. The number of edges in $\mathcal{M}$ is called the *cardinality* of the matching and a matching for G_A or G_B is referred to as perfect one if its edges touch all vertices. We refer to [10] and the reference therein for conditions which guarantee the existence of perfect matching. A perfect matching $\mathcal{M}$ for G_A or G_B corresponds to n nonzeros no two of which are in the same row or column and can be represented in terms of a column permutation

$$\pi_{ji} = \begin{cases} 1, & \text{if } (i, j) \in \mathcal{M} \\ 0, & \text{otherwise} \end{cases}$$

such that the matrix $A\pi$ has a zero-free diagonal. Generally, in linear algebra applications, we are interested in finding matching that controls the size of the diagonal elements of $A\pi$, and such a requirement is formulated in terms of maximum weighted bipartite matching problem. In particular, matrices with larger entries on the diagonal can be obtained by solving the following optimization problem.

– **Maximum Product Bipartite Matching Problem:** Given graph G_B corresponding to a sparse matrix A, find a matching $\mathcal{M}$ that maximizes the product of the matched entries, i.e., find a permutation matrix π such that $\prod abs((A\pi)_{ii})$ is maximum among all permutations.

Therefore, if row i is matched to column j in a maximum product bipartite matching problem, we can conclude that $|a_{ij}| \approx max_{k\neq i}|a_{ik}|$, which in terms of the classical AMG characterization of the strength of matrix connections is equivalent to say that index i is strongly connected to index j. The difference is that the maximum product bipartite matching problem optimizes a global measure, whereas in classical AMG the strength of connection is a local notion. We demonstrate in the present paper that this global matching is able to capture very accurately the direction of strong anisotropy for difficult AMG test problems with anisotropy that is not grid-aligned. We note however that the maximum product bipartite matching problem if implemented exactly can become too costly, on the other hand a similar matching problem can be described for undirected graphs, so in practice we use approximation of the maximum product matching problem in undirected graph to end up with setup cost of order $\mathcal{O}(n)$ and still be able to capture the direction of strong anisotropy as in the more expensive accurate solution of the maximum product bipartite matching problem.

Motivated by the above considerations, we propose a coarsening process based on the pairwise aggregation described in Algorithm 1. It builds a partition $\mathfrak{a}_k,\ k=1,\dots,n_c$ of the index set $\{1,\dots,n\}$, where each aggregate $\mathfrak{a}_k$ is generally a pair of matched indices. In the general case of possible unmatched indices, i.e., in the case of non-perfect matching (structurally rank-deficient matrices) or sub-optimal solutions, we can obtain a partition with possible singletons.

We observe that Algorithm 1 is an automatic aggregation procedure only using information on matrix entries and it does not depend on any user-defined strong/weakly connection threshold. Computation of a maximum product perfect matching of a graph is a challenging problem in terms of computational complexity, indeed classical algorithms require a running time of $O(n^3)$ [4]. On the other hand, the problem can be solved for bipartite graphs with the widely used algorithm described in [9] and implemented in the `HSL-MC64` subroutine [13], whose computational complexity is $O(n(nnz+n)\log n)$, where nnz is the number of nonzeros of the matrix. The latter cost is a worst case estimate. At any

```
**Data**: matrix $A$ of dimension $n$
**Result**: $n_p$, $n_s$, $n_c$ and sets of aggregates $\mathfrak{a}_1,\dots \mathfrak{a}_{n_c}$
compute $\mathcal{M}$ **weighted matching** for $A$;
$n_c = 0$, $n_p = 0$, $n_s = 0$;
$U = [1,\dots,n]$;
**while** $U \neq \emptyset$ **do**
    Pick an $i \in U$;
    **if** $\exists j \in U \setminus \{i\}$ *such that* $(i,j) \in \mathcal{M}$ **then**
        $n_p = n_p + 1$;
        $n_c = n_c + 1$;
        $\mathfrak{a}_{n_c} = \{i,j\}$;
        $U = U \setminus \{i,j\}$;
    **else**
        $n_s = n_s + 1$;
        $n_c = n_c + 1$;
        $\mathfrak{a}_{n_c} = \{i\}$;
        $U = U \setminus \{i\}$;
    **end**
**end**
```

Algorithm 1: Pairwise aggregation based on weighted matching

rate, from AMG perspective the latter cost is still unacceptable since our ultimate goal is aiming at $\mathcal{O}(n)$ algorithm. For that reason, we also use an approximate version of a maximum weighted matching algorithm in an undirected graph that uses $\mathcal{O}(n)$ operations. We demonstrated that, although in the case of approximate matching, the coarsening ratio of our approach is reduced with respect to the coarsening by a factor of two in the exact perfect matching, the overall performance of the adaptive process does not deteriorate substantially.

3 Coarsening based on compatible weighted matching

3.1 Main ingredients for coarsening

Given a set of aggregates $\mathfrak{a}_1,\dots \mathfrak{a}_{n_c}$, built by Algorithm 1, and a starting (arbitrary) vector $\mathbf{w}$, per each pair $\mathfrak{a}_l=\{i,j\},\ l=1,\dots,n_p$, let

$$\mathbf{w}_{\mathfrak{a}_l} = \frac{1}{\sqrt{w_i^2+w_j^2}}\begin{bmatrix} w_i \\ w_j \end{bmatrix},\quad \mathbf{w}_{\mathfrak{a}_l}^{\perp} = \frac{1}{\sqrt{w_i^2+w_j^2}}\begin{bmatrix} -w_j \\ w_i \end{bmatrix}$$

be the normalized restrictions of $\mathbf{w}$ to the set $\mathfrak{a}_l$ and its orthonormal complement. We then define the following matrices:

$$\tilde{P}_c = \text{blockdiag}(\mathbf{w}_{\mathfrak{a}_1},\dots,\mathbf{w}_{\mathfrak{a}_{n_p}}) \in \mathbb{R}^{2n_p\times n_p},$$

$$\tilde{P}_f = \text{blockdiag}(\mathbf{w}_{\mathfrak{a}_1}^{\perp},\dots,\mathbf{w}_{\mathfrak{a}_{n_p}}^{\perp}) \in \mathbb{R}^{2n_p\times n_p}.$$

For the singletons $\mathfrak{a}_l=\{k\},\ l=1,\dots,n_s$, $(n_c=n_p+n_s,\ n=2n_p+n_s)$, we introduce the diagonal matrix:

$$W = diag(w_k/|w_k|) \in \mathbb{R}^{n_s\times n_s}.$$

From the above matrices, we obtain two prolongation matrices corresponding to two complementary coarse index sets:

$$P_c = \begin{pmatrix} \tilde{P}_c & 0 \\ 0 & W \end{pmatrix} \in \mathbb{R}^{n \times n_c}, \quad P_f = \begin{pmatrix} \tilde{P}_f \\ 0 \end{pmatrix} \in \mathbb{R}^{n \times n_p}. \quad (3.1)$$

The $n \times n_c$ matrix P_c, referred to as *tentative prolongator*, maps vectors associated with the coarse index set $\{1, 2, \dots, n_c\}$ on the original fine-grid set $\{1, 2, \dots, n\}$, whereas P_f, referred to as *complementary tentative prolongator*, is an $n \times n_p$ matrix which transfers vectors associated with the complementary coarse index set $\{1, 2, \dots, n_p\}$ also on the fine-grid index set $\{1, 2, \dots, n\}$. We recall that $n_c = n_p + n_s$ and $n = 2n_p + n_s$, where n_p is the number of pairwise aggregates and n_s is the number of singletons. Note that $\mathbb{R}^n = \text{Range}(P_c) \oplus^{\perp} \text{Range}(P_f)$, where $\text{Range}(P_c) \ni \mathbf{w}$ and $\text{Range}(P_f) \ni \mathbf{w}^{\perp}$ form an orthogonal decomposition of $\mathbb{R}^n$. In other words, we have that the matrix $P = [P_f, \ P_c]$ is orthogonal.

After proper reordering of A, the following two coarser matrices can be formed via the Galerkin triple matrix products

$$A_c = P_c^T A P_c \in \mathbb{R}^{n_c \times n_c}, \qquad A_f = P_f^T A P_f \in \mathbb{R}^{n_p \times n_p}. \quad (3.2)$$

These are the diagonal blocks of the transformed fine-grid matrix $P^T AP$ under the orthogonal transformation P, i.e., we have

$$P^T AP = \begin{bmatrix} A_f & A_{fc} \\ A_{cf} & A_c \end{bmatrix}.$$

The off-diagonal blocks read: $A_{fc} = P_f^T A P_c$ and $A_{cf} = P_c^T A P_f$.

The choice of the best coarse matrix A_c for a multilevel hierarchy can be driven by the basic principle of compatible relaxation first introduced by Brandt in [1] and extended in [11] (see also [21]). The compatible relaxation is defined as a relaxation scheme which is able to keep coarse-level variables invariant. It gives a practical way to measure the quality of a set of coarse variables, indeed, since in an efficient multigrid method relaxation scheme has to be effective on the fine variables, the convergence rate of a compatible relaxation scheme can be used as a measure of the quality of a set of coarse variables. This basic idea was used in different approaches to select coarse grids [6,15]. Here, we apply the principle of compatible relaxation to choose the best coarse matrix from the two available matrices in (3.2), and the corresponding coarse index set, by applying a simple point-wise relaxation scheme to the homogeneous systems associated to each of the matrices, starting from a random initial guess and then relaxing on the two complementary vector spaces separately. If the vector $\mathbf{w}$ is chosen based on a relaxation scheme applied to the original matrix A so that it is in the near-null space of A, it is natural to expect that A_f will be better conditioned than A_c. For a more general iterative process, we allow the option to choose between A_f and A_c when selecting the coarse-level variables.

3.2 The multilevel adaptive coarsening schemes

Our overall adaptive multilevel coarsening strategy can be described as follows. We propose two versions. The first one, referred to as *coarsening based on compatible matching, rel.1* is sketched in Algorithm 2. We start with the given system matrix and a given smooth vector, for example the unitary vector. Then, we apply Algorithm 1 for building the two complementary coarse matrices in (3.2). After that, we test the convergence of a simple smoother on homogeneous systems associated with the two available matrices and choose as new coarse matrix and new algebraically smooth vector those for which slower convergence is observed. The process can be applied in a recursive way until a desired small size of the coarse matrix is obtained. Therefore, our procedure builds a binary tree of multiple coarse spaces by matching-based aggregation, where, at each level, selection of the new coarsening branch is based on compatible relaxation of a given vector.

```
Data: $A$ matrix, $\mathbf{w}$ (smooth) vector, $maxsize$ maximum
      size for the coarsest matrix
Result: hierarchy of coarse matrices $A^k$ (and intergrid
        operators)
$A^1 = A$, $k = 1$;
relax $\nu_1$ times on $A^1\mathbf{w}^1 = \mathbf{0}$ starting with $\mathbf{w}$;
while $size(A^k) > maxsize$ do
    compute partition $\mathfrak{a}_l$ by Algorithm 1 applied to
    $A^k - diag(A^k)$;
    build $A_c^k$ and $A_f^k$ from $\mathfrak{a}_l$ and $\mathbf{w}^k$;
    relax $\nu_1$ times on $A_c^k\mathbf{w}_c = \mathbf{0}$ and on $A_f^k\mathbf{w}_f = \mathbf{0}$
    starting with a random guess;
    estimate convergence rates $\rho_f$ and $\rho_c$;
    if $\rho_f < \rho_c$ then
        $A^{k+1} = A_c^k$, $\mathbf{w}^{k+1} = \mathbf{w}_c$;
    else
        $A^{k+1} = A_f^k$, $\mathbf{w}^{k+1} = \mathbf{w}_f$;
    end
    $k = k + 1$;
end
```
Algorithm 2: Coarsening based on compatible matching, rel. 1

Note that, as shown in [17], in the case of strongly diagonally dominant or s.p.d. matrices maximum product matching produces permutation matrices equal to the identity matrices, i.e. it produces a set of n self-aggregated indices. Therefore, in order to obtain an effective pairwise aggregation, in Algorithm 2, we apply the maximum product matching to the matrix $A^k - \mathrm{diag}(A^k)$, where $\mathrm{diag}(A^k)$ is the diagonal matrix obtained by the diagonal elements of A^k. We also observe that in Algorithm 2, when we build the two complementary coarse matrices A_c and A_f, we need to compute the normalized restriction of the smooth vector $\mathbf{w}$ on each set of the partition computed by Algorithm 1. It may happen that during the coarsening process, the smooth vector components corresponding to some set of the partition are very small, i.e. the corresponding error components are sufficiently damped by the smoother. In these cases we associate the corresponding unknowns to the vector space $\mathrm{Range}(P_f)$. Convergence rates in Algorithm 2 can be estimated as the ratios of the A-norm of two successive iterates, that is $\rho_c = \|\mathbf{w}_c^k\|_{A_c} / \|\mathbf{w}_c^{k-1}\|_{A_c}$ and $\rho_f = \|\mathbf{w}_f^k\|_{A_f} / \|\mathbf{w}_f^{k-1}\|_{A_f}$.

There is an alternative to Algorithm 2 that we consider, still using both the orthogonal decomposition of $\mathbb{R}^n$ defined by the matrices in (3.1) and the principles of compatible relaxation to build an effective coarsening process. Indeed, after we have built the matrices in (3.2), we select A_c as the coarse matrix if the corresponding complementary matrix A_f is diagonally-dominant, i.e., if A_f has the compatible relaxation fast to converge. We observe that given the original matrix A, its associated graph G_A, and a vector $\mathbf{w}$, the diagonal entries of the resulting A_f are a subset of the following values:

$$\widehat{a}_{i,j} = \frac{1}{w_j^2 + w_i^2} \begin{bmatrix} -w_j \\ w_i \end{bmatrix}^T \begin{pmatrix} a_{i,i} & a_{i,j} \\ a_{j,i} & a_{j,j} \end{pmatrix} \begin{bmatrix} -w_j \\ w_i \end{bmatrix}, \ (i,j) \in E. \tag{3.3}$$

Consider the thus modified symmetric matrix $\widehat{A} = \widehat{a}_{i,j} \in \mathbb{R}^{n \times n}$ having a null diagonal and the same sparsity pattern as A. Note that building $\widehat{A}$ has a computational cost of $\mathcal{O}(nnz)$. Then, if we compute a maximum product weighted matching $\mathcal{M} \subseteq E$ from $\widehat{A}$ and build the corresponding aggregates, we see that the complementary tentative prolongator P_f in (3.1) produces a matrix A_f which has on its diagonal entries $\widehat{a}_{i,j}$, $(i,j) \in \mathcal{M}$ with maximal product. The latter can be seen as an approximation to the notion of diagonal dominance giving rise to a fast convergent compatible relaxation. The process can be applied in a recursive way to define a new adaptive coarsening algorithm which we refer to as *coarsening based on compatible matching, rel 2.* It is sketched in Algorithm 3. Note that also in this algorithm, at each level possible small smooth vector entries are not associated to any coarse unknown.

```
Data: $A$ matrix, $\mathbf{w}$ (smooth) vector, $maxsize$ maximum
      size for the coarsest matrix
Result: hierarchy of coarse matrices $A^k$ (and intergrid
        operators)
$A^1 = A$, $k = 1$;
relax $\nu_1$ times on $A^1\mathbf{w}^1 = \mathbf{0}$ starting with $\mathbf{w}$;
while $size(A^k) > maxsize$ do
    build $\widehat{A}^k$ from $A^k$ and $\mathbf{w}^k$;
    compute partition $\mathfrak{a}_l$ by Algorithm 1 applied to $\widehat{A}^k$;
    build $A_c^k$ from $\mathfrak{a}_l$ and $\mathbf{w}^k$;
    relax $\nu_1$ times on $A_c^k\mathbf{w}_c = \mathbf{0}$ starting with a random
    guess;
    $A^{k+1} = A_c^k$, $\mathbf{w}^{k+1} = \mathbf{w}_c$;
    $k = k + 1$;
end
```

Algorithm 3: Coarsening based on compatible matching, rel. 2

The above two compatible matching based coarsening algorithms can be used to define a hierarchy of coarse vector spaces and matrices from which a multilevel method B can be designed. In the following, we describe an adaptive strategy to improve the efficiency of an initial multilevel method obtained with compatible matching based coarsening by successively building a composite method with a prescribed convergence rate.

4 Composite AMG with prescribed convergence rate

Following the αAMG methodology, once an algebraic multilevel solver B has been constructed, we test its performance by solving the homogeneous problem $A\mathbf{x} = \mathbf{0}$, i.e. by performing the following iterations:

$$\mathbf{x}_k = (I - B^{-1}A)\mathbf{x}_{k-1}, \qquad k = 1, 2, \ldots,$$

starting with a random initial iterate $\mathbf{x}_0$ and monitoring convergence through two successive values of the A-norm of the error (which is equal to the respective iterate, since the exact solution is zero). The above iterates provide approximation to the lowest eigenmode of $B^{-1}A$, which is commonly referred to as algebraic smooth vectors with respect to the current AMG method. If the convergence factor of the method is close to one, we can select $\mathbf{w} = \mathbf{x}_k / \|\mathbf{x}_k\|_A$ and apply one of the coarsening algorithms described in the preceding section to generate a new method B_1 based on this new vector $\mathbf{w}$. Assuming that we have constructed two (or more) methods B_r, $r = 0, 1, ..., m$ via the above bootstrap scheme aimed at improving the initial AMG, we consider the homogeneous system and monitor the convergence of the following composite method, starting with a random initial guess $\mathbf{x}_0$,

$$\mathbf{x}_k = \prod_{r=1}^{m} (I - B_r^{-1} A)\mathbf{x}_{k-1}, \qquad k = 1, 2, \ldots, \tag{4.1}$$

or of its symmetrized version:

$$\mathbf{x}_k = \prod_{r=0}^{2m+1} (I - B_r^{-1} A)\mathbf{x}_{k-1}, \qquad k = 1, 2, \ldots, \tag{4.2}$$

where $B_{m+r} = B_{m+1-r},\ r = 1, \ldots, m+1$. The process may be repeated by computing at each stage a new multilevel method until the convergence rate of the composite AMG is acceptable. The final adaptive procedure is sketched in Algorithm 4.

```
Building Phase: build a new AMG component
    1. let $m = 1$ and $\mathbf{w}^1$ (be an initial vector, e.g. $\mathbf{w}^1 = \mathbf{1}$);
    2. apply Algorithm 2 or Algorithm 3 to $A$ and $\mathbf{w}^m$
       for building hierarchy of coarser matrices $A^k$ and
       prolongators $P^k$;
    3. define $B_m$ as a standard (V, W, or FM)-cycle based
       on the new hierarchy;
Testing Phase: apply the composite AMG and expose (further)
  smooth errors
    3. let $\mathbf{x}_0$ a random vector;
    4. apply iterations (4.1) or (4.2) for $\nu_2$ times
       on $A\mathbf{x} = \mathbf{0}$;
    5. estimate convergence rate $\rho$ of the composite AMG;
    6. if $\rho > \rho_{desired}$, set $\mathbf{w}^{m+1} = \mathbf{x}_{\nu_2}/\|\mathbf{x}_{\nu_2}\|_A$,
       $m = m + 1$, go to 2.
```

Algorithm 4: Composite αAMG - Setup Phase

5 Results

In this Section we illustrate the performance of our composite αAMG in terms of the cost of the setup phase described in Algorithm 4 and the ability of the coarsening procedures based on maximum product matching to obtain effective coarse grids.

We considered the following anisotropic PDE posed in the unit square, when homogeneous Dirichlet boundary conditions are considered:

$$-\text{div}(K\ \nabla u) = f,$$

where K is the coefficient matrix

$$K = \begin{bmatrix} a & c \\ c & b \end{bmatrix}, \qquad \text{with} \qquad \begin{cases} a = \epsilon + \cos^2(\theta) \\ b = \epsilon + \sin^2(\theta) \\ c = \cos(\alpha)\sin(\theta) \end{cases}$$

The parameter $0 < \epsilon \leq 1$ defines the strength of anisotropy in the problem, while the parameter θ specifies the direction of anisotropy. In the following we discuss results related to $\epsilon = 0.001$ and $\theta = 0,\ \pi/8,\ \pi/4,\ \pi/3,\ \pi/2$ for a total of 5 test cases, which we refer to as *Test Case* 1 to 5, respectively. The above problem was discretized by the Matlab PDE toolbox, using bilinear finite elements on triangular and rectangular meshes.

We measure the setup cost in terms of AMG components (*stages*) built by the adaptive process in Algorithm 4, both in the case of the coarsening described in Algorithm 2 and in the case of Algorithm 3. In addition to the number of the components, we also report, per each test case and per each mesh, the convergence factor (*rho*) of the composite solver, the average number of levels (*nlev*) of all built solver components and the average of their operator complexity (*cmpx*). This last parameter is commonly defined as the ratio between the sum of nonzero entries of the matrices of all levels and the number of nonzero entries of the fine matrix; it gives an estimate of the cost of application of a cycle. Many algorithmic and parameter choices are possible to test our method; here we discuss results related to the following particular choices. The desired convergence factor required for the composite AMG was set to $\rho_{desired} = 0.7$ and a symmetrized multiplicative composition of the AMG components as in (4.2) was applied. The number of iterations used to estimate solver convergence rates at each stage was set to $\nu_2 = 15$. Weighted Jacobi was applied as relaxation scheme in Algorithm 2 and Algorithm 3, where we have fixed the number of iterations equal to $\nu_1 = 20$. We stop the coarsening process when the size of the coarsest matrix was at most maxsize = 100. Note that we did various experiments with increased values of ν_1 and ν_2 but estimated values of the obtained convergence rates did not differ significantly.

We developed a Matlab implementation of the composite αAMG and we analyze its behavior when the coarsening algorithm is based on algorithm `HSL-MC64` (Section 5.1), or based on a Matlab implementation of the linear complexity half-approximation maximum weighted matching algorithm for undirected graphs described in [19] (Section 5.2).

5.1 Composite AMG based on exact matching

In the following we analyze the setup cost of our bootstrap strategy for building a composite multigrid of type 4.2, when each AMG component was a W(1,1)-cycle, with one sweep of symmetric Gauss-Seidel used as both pre/post smoother and as coarsest level solver. Here we discuss results obtained using the `HSL-MC64` routine which, for non-singular matrices, is able to compute a perfect weighted matching for bipartite sparse matrix graphs. In this case, Algorithm 1 has a coarsening factor less but close to two, since it can produce a (small) number of singletons (unaggregated DOFs), essentially due to possible unsymmetric matching (e.g. row i matched at column j and row j matched at column k, with $k \neq i$).

5.1.1 Unstructured mesh

In this Section we present results for matrices corresponding to discretization of our test cases on unstructured triangular meshes with a total number of nodes $n = 2705, 10657, 42305$, that correspond to three different mesh sizes. We report, in the first five pairs of tables (Tables 5.1–5.5), all parameters leading to the setup cost of the composite AMG achieving convergence rate not larger than the prescribed one, $\rho_{desired} = 0.7$.

Composite αAMG Setup, coarsening based on Algorithm 2

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	1	0.69	6	1.96
10675	3	0.63	8	1.98
42305	6	0.66	10	1.99

Composite αAMG Setup, coarsening based on Algorithm 3

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	1	0.69	6	1.94
10657	3	0.64	8	1.98
42305	5	0.69	11	2.00

Table 5.1 *Test Case 1: ($\theta = 0$).* Setup cost for different mesh sizes when exact bipartite matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	1	0.29	6	1.96
10675	1	0.38	8	1.99
42305	1	0.50	10	2.00

Composite αAMG Setup, coarsening based on Algorithm 3

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	1	0.29	6	1.95
10657	1	0.36	8	1.99
42305	1	0.50	10	2.00

Table 5.2 *Test Case 2: ($\theta = \pi/8$).* Setup cost for different mesh sizes when exact bipartite matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	1	0.69	6	1.93
10675	3	0.68	8	1.98
42305	6	0.68	10	2.00

Composite αAMG Setup, coarsening based on Algorithm 3

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	2	0.58	6	1.94
10657	3	0.67	8	1.98
42305	5	0.70	10	2.00

Table 5.3 *Test Case 3: ($\theta = \pi/4$).* Setup cost for different mesh sizes when exact bipartite matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	1	0.70	6	1.95
10675	3	0.63	8	1.99
42305	6	0.65	10	2.00

Composite αAMG Setup, coarsening based on Algorithm 3

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	2	0.51	6	1.94
10657	3	0.63	8	1.98
42305	6	0.70	10	2.00

Table 5.4 *Test Case 4: ($\theta = \pi/3$).* Setup cost for different mesh sizes when exact bipartite matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	1	0.70	6	1.96
10675	2	0.70	8	1.99
42305	5	0.67	10	2.0

Composite αAMG Setup, coarsening based on Algorithm 3

n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	1	0.70	6	1.95
10657	3	0.59	8	1.99
42305	5	0.69	10	2.00

Table 5.5 *Test Case 5: ($\theta = \pi/2$).* Setup cost for different mesh sizes when exact bipartite matching is used for aggregation.

We can see that in all the cases our method, for both coarsening algorithms (Algorithm 2 and Algorithm 3), shows very similar results and it is able to achieve a convergence factor less than the desired one with a moderate number of components (denoted as *nstages* in the tables). This demonstrates feasibility and robustness of our approach. If we look closer at the convergence behavior in the different test cases, we can observe that the method shows very good efficiency and scalability on *Test Case 2*, where a convergence rate much lower than the required one is obtained, for all mesh sizes, by building only 1 AMG component. An increase in the number of coarsening levels corresponding to increased mesh size produces only a slight degradation in the convergence rate of the solver. In all other test cases, the convergence behavior appears mesh dependent, showing an increase of the number of solver components when the mesh gets refined. Indeed, in all cases with the exception of *Test Case 2*, we need 5 or 6 components to get the desired convergence rate for the finest mesh versus 1 or 2 components needed in the case of the smallest mesh size. In all cases the average operator complexity over all constructed solver components is about two.

Concerning the performance of the coarsening process, we observe that both versions of the compatible weighted bipartite matching generate similar coarsening trees. More specifically, we see that Algorithm 2 based on an adaptive choice of the coarsening tree branch which depends on the convergence behavior of the relaxation scheme applied to two orthogonal vector spaces always chooses (at each level) the tree branch associated with the matrix A_c. This shows that the pairwise aggregation algorithm based on maximum product matching of the original system matrix (that is A, not the modified one, $\widehat{A}$) is able to detect strong matrix connections (since then A_f has faster to converge compatible relaxation) for our test cases. In Figures 5.1 and 5.2, we can see that the estimated convergence factors ρ_c and ρ_f of the compatible relaxation applied to the matrices A_c and A_f respectively produced by our two coarsening schemes, Algorithm 2 and Algorithm 3, have very similar pattern.

$$\begin{tikzcd}
 & & & & & A^1 \arrow[dl, "\varrho_c{=}0.98"'] \arrow[dr, "\varrho_f{=}0.81"] & \\
 & & & & A^1_c = A^2 \arrow[dl, "\varrho_c{=}0.98"'] \arrow[dr, "\varrho_f{=}0.79"] & & A^1_f \\
 & & & A^2_c = A^3 \arrow[dl, "\varrho_c{=}0.99"'] \arrow[dr, "\varrho_f{=}0.78"] & & A^2_f & \\
 & & A^3_c = A^4 \arrow[dl, "\varrho_c{=}0.99"'] \arrow[dr, "\varrho_f{=}0.71"] & & A^3_f & & \\
 & A^4_c = A^5 \arrow[dl, "\varrho_c{=}0.99"'] \arrow[dr, "\varrho_f{=}0.74"] & & A^4_f & & & \\
A^5_c = A^6 & & A^5_f & & & &
\end{tikzcd}$$

Fig. 5.1 *Test Case 5 ($\theta = \pi/2$)*, $n = 2705$. Coarsening tree based on Algorithm 2 and exact bipartite matching.

The coarsening trees depicted in Figs. 5.1-5.2 are representative of the behavior of the coarsening process for each component of the composite αAMG solvers built for all considered test cases and various mesh sizes.

$$\begin{tikzcd}
 & & & & & A^1 \arrow[dl, "\varrho_c{=}0.98"'] \arrow[dr, "\varrho_f{=}0.82"] & \\
 & & & & A^1_c = A^2 \arrow[dl, "\varrho_c{=}0.99"'] \arrow[dr, "\varrho_f{=}0.79"] & & A^1_f \\
 & & & A^2_c = A^3 \arrow[dl, "\varrho_c{=}0.99"'] \arrow[dr, "\varrho_f{=}0.79"] & & A^2_f & \\
 & & A^3_c = A^4 \arrow[dl, "\varrho_c{=}0.99"'] \arrow[dr, "\varrho_f{=}0.77"] & & A^3_f & & \\
 & A^4_c = A^5 \arrow[dl, "\varrho_c{=}0.99"'] \arrow[dr, "\varrho_f{=}0.77"] & & A^4_f & & & \\
A^5_c = A^6 & & A^5_f & & & &
\end{tikzcd}$$

Fig. 5.2 *Test Case 5 ($\theta = \pi/2$)*, $n = 2705$. Coarsening tree based on Algorithm 3 and exact bipartite matching.

More specifically, we observe that the estimated convergence factor of the compatible relaxation, that is, of the weighted Jacobi applied to the homogeneous system associated with A_f built at each coarsening level, decreases fairly when the number of levels increases and stays within the range $[0.71, 0.85]$ for all tested mesh sizes. Such bounded convergence rates of the compatible relaxation when the number of levels and the problem size increase are good indication that our two coarsening schemes are capable of producing scalable AMG. In the following figures, Figs. 5.3-5.4, we show illustration of the pattern of the aggregates (or sparsity of the interpolation matrices) built by our two coarsening algorithms for two different test cases corresponding to the smallest problem size. Note that points corresponding to fine grid are represented in the figures by symbol $+$ in black, while orange lines and boxes represent aggregates built at the coarsest level. The number of aggregates at the coarsest level is $n_c = 93$ for both pictures in Fig. 5.3, while in Fig. 5.4 we have $n_c = 92$ for the picture on the top, corresponding to Algorithm 2, and $n_c = 91$ and $n_c = 92$, for the pictures in the middle and on the bottom, respectively, corresponding to the 2-stages AMG built when Algorithm 3 is applied.

In Figures 5.3 and 5.4 we can clearly see that both coarsening algorithms were able to produce *semi-coarsening* which detects the direction of anisotropy by building aggregates aligned with the $x-$direction for *Test Case 1* and with the main diagonal for *Test Case 3.*

5.1.2 Structured mesh

In this subsection we report results for linear systems arising from three of the test cases presented in the previous subsection, corresponding to $\theta = 0$, $\pi/4$ and $\pi/3$, now using rectangular mesh with increasing number of

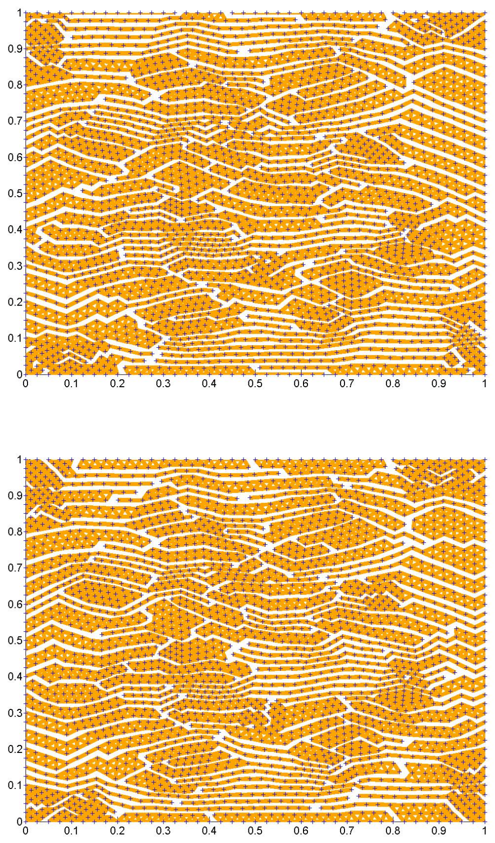

Fig. 5.3 *Test case 1 ($\theta = 0$)*, $n = 2705$. Coarsest interpolation matrices pattern built by Algorithm 2 (top) and Algorithm 3 (bottom) with exact matching.

nodes in the discretization. The goal is to demonstrate that our coarsening algorithms can easily detect grid-aligned anisotropy (cases $\theta = 0$ and $\pi/4$) and after some additional work, the adaptive procedure can produce semi-coarsening also in the non-grid aligned anisotropic case ($\pi/3$). This is indeed the case and is illustrated in Figs. 5.7-5.8-5.9 for a mesh with 40 internal nodes per each direction, where the number of aggregates at the coarsest level are $n_c = 58$ for both the pictures of Fig. 5.7; $n_c = 63$ and $n_c = 60$ for the pictures at the top and the bottom of Fig. 5.8, respectively; $n_c = 56$ and $n_c = 58$ for the pictures at the top and the bottom of Fig. 5.9, respectively. Note that in Figure 5.8, at the top left and bottom right, black bullets correspond to nodes not aggregated due to near zero smooth error at these points obtained after relaxation.

The parameter setting to construct the solver, the smoother and the algorithmic choices are the same as in the previous unstructured mesh case.

We first note that, as in the case of unstructured meshes, the two coarsening processes give similar results for all the test cases. In terms of setup cost, we

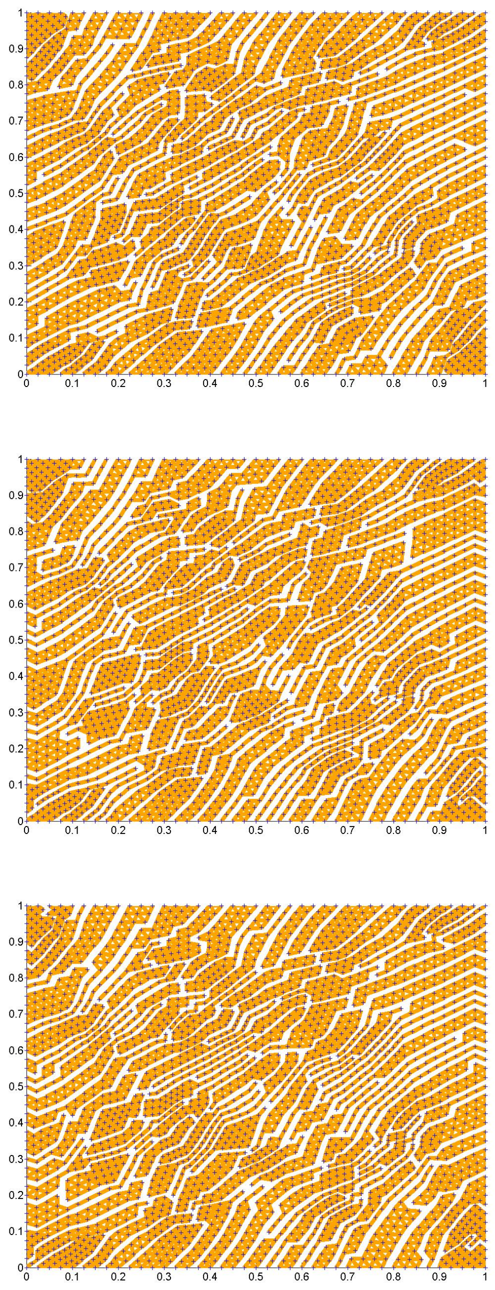

Fig. 5.4 *Test case 3 ($\theta = \pi/4$)*, $n = 2705$. Coarsest interpolation matrices pattern built by Algorithm 2 (top) and Algorithm 3 (center and bottom) with exact bipartite matching.

observe that in the easy case of grid-aligned anisotropy, *Test Case 1* and *Test Case 3*, only 1 or at most 2 components are needed to achieve convergence factor not greater than the desired one showing a very good scalability of the method. On the other hand, as in the unstructured grid case, for *Test Case 4* when the anisotropy is not grid-aligned, we observe a degradation of the scalability, i.e., the number of components

Composite αAMG Setup, coarsening based on Algorithm 2				
n	*nstages*	ρ	*nlev*	*cmpx*
64×64	1	0.63	7	2.0
128×128	1	0.70	9	2.1
256×256	2	0.61	11	2.1

Composite αAMG Setup, coarsening based on Algorithm 3				
n	*nstages*	ρ	*nlev*	*cmpx*
64×64	1	0.40	7	2.0
128×128	1	0.63	9	2.1
256×256	2	0.61	11	2.1

Table 5.6 *Test Case 1: ($\theta = 0$).* Setup cost for different mesh sizes when exact bipartite matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2				
n	*nstages*	ρ	*nlev*	*cmpx*
64×64	1	0.36	7	1.95
128×128	1	0.55	9	1.97
256×256	2	0.56	11	1.98

Composite αAMG Setup, coarsening based on Algorithm 3				
n	*nstages*	ρ	*nlev*	*cmpx*
64×64	1	0.28	7	1.95
128×128	1	0.53	9	1.97
256×256	2	0.56	11	1.99

Table 5.7 *Test case 3: ($\theta = \pi/4$).* Setup cost for different mesh sizes when exact bipartite matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2				
n	*nstages*	ρ	*nlev*	*cmpx*
64×64	2	0.66	7	1.97
128×128	4	0.68	9	2.00
256×256	8	0.67	11	2.00

Composite αAMG Setup, coarsening based on Algorithm 3				
n	*nstages*	ρ	*nlev*	*cmpx*
64×64	2	0.62	7	1.96
128×128	4	0.64	8	1.98
256×256	7	0.67	11	1.99

Table 5.8 *Test case 4: ($\theta = \pi/3$).* Setup cost for different mesh sizes when exact bipartite matching is used for aggregation.

needed to reach the desired convergence factor increases when the mesh is refined. Also for these test cases the average operator complexity is about two for each mesh size similarly to the unstructured mesh case. As in the unstructured mesh case described in the previous section, we observe that the behavior of the coarsening process is very similar per each AMG component of the composite solver. It also appears comparable to that obtained for the same test cases arising from discretization on unstructured grids and discussed before, although here we observed almost constant convergence rate of the compatible relaxation (≈ 0.8) at each level of the coarsening tree, for all test cases and each mesh. As representatives of the general behavior, we draw in Figures 5.5 and 5.6 the coarsening trees built by two versions of our matching-based coarsening methods for the first component of the 2-stage composite AMG for *Test case 4* using the smallest size structured mesh.

$$\begin{tikzcd}
& & & & & & A^1 \arrow[dl, "\varrho_c=0.97"'] \arrow[dr, "\varrho_f=0.80"] & \\
& & & & & A_c^1 = A^2 \arrow[dl, "\varrho_c=0.98"'] \arrow[dr, "\varrho_f=0.79"] & & A_f^1 \\
& & & & A_c^2 = A^3 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.77"] & & A_f^2 & \\
& & & A_c^3 = A^4 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.76"] & & A_f^3 & & \\
& & A_c^4 = A^5 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.76"] & & A_f^4 & & & \\
& A_c^5 = A^6 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.77"] & & A_f^5 & & & & \\
A_c^6 = A^7 & & A_f^6 & & & & &
\end{tikzcd}$$

Fig. 5.5 *Test case 4 ($\theta = \pi/3$), $n = 64 \times 64$.* Coarsening tree based on Algorithm 2 and exact bipartite matching.

$$\begin{tikzcd}
& & & & & & A^1 \arrow[dl, "\varrho_c=0.98"'] \arrow[dr, "\varrho_f=0.80"] & \\
& & & & & A_c^1 = A^2 \arrow[dl, "\varrho_c=0.98"'] \arrow[dr, "\varrho_f=0.84"] & & A_f^1 \\
& & & & A_c^2 = A^3 \arrow[dl, "\varrho_c=0.98"'] \arrow[dr, "\varrho_f=0.83"] & & A_f^2 & \\
& & & A_c^3 = A^4 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.82"] & & A_f^3 & & \\
& & A_c^4 = A^5 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.80"] & & A_f^4 & & & \\
& A_c^5 = A^6 \arrow[dl, "\varrho_c=0.98"'] \arrow[dr, "\varrho_f=0.80"] & & A_f^5 & & & & \\
A_c^6 = A^7 & & A_f^6 & & & & &
\end{tikzcd}$$

Fig. 5.6 *Test case 4 ($\theta = \pi/3$), $n = 64 \times 64$.* Coarsening tree based on Algorithm 3 and exact bipartite matching.

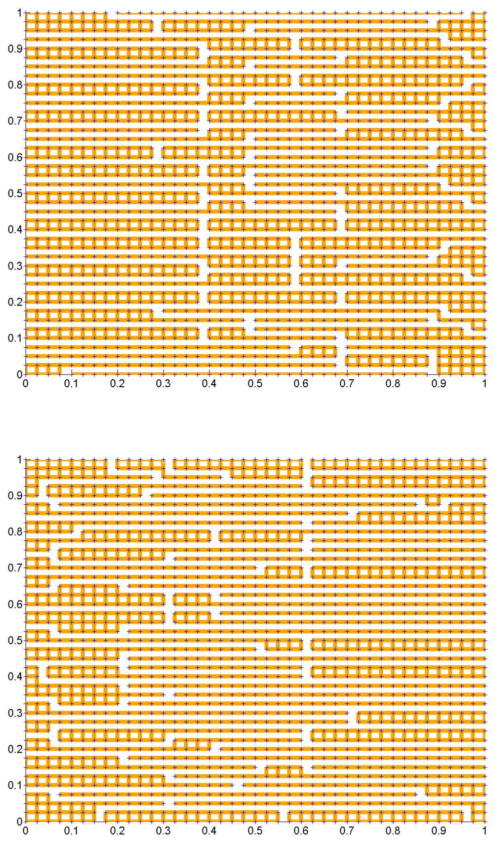

Fig. 5.7 *Test Case 1 ($\theta = 0$)*, $n = 40{\times}40$. Coarsest interpolation matrices pattern built by Algorithm 2 (top) and Algorithm 3 (bottom) with exact bipartite matching.

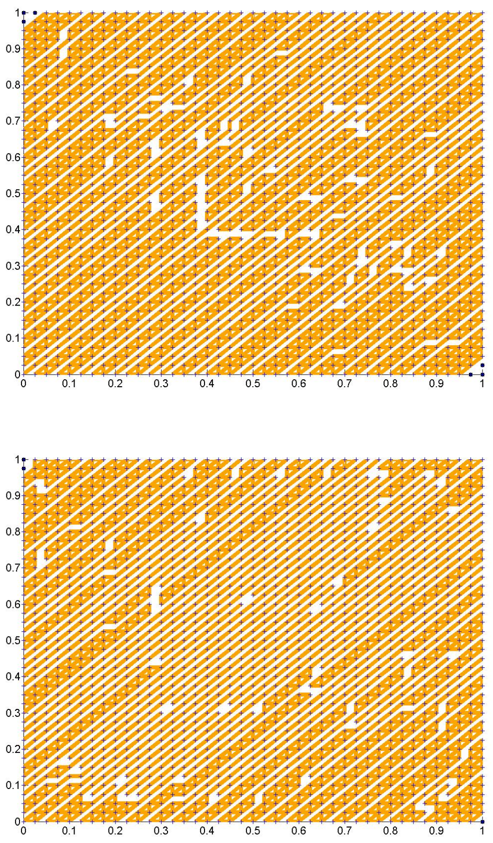

Fig. 5.8 *Test Case 3 ($\theta = \pi/4$)*, $n = 40 \times 40$. Coarsest interpolation matrices pattern built by Algorithm 2 (top) and Algorithm 3 (bottom) with exact bipartite matching.

5.2 Composite AMG based on approximate matching

As we remark at the end of Section 2, the `HSL-MC64` subroutine used for computing a maximum product matching in a bipartite graph has a non-optimal computational complexity. This is not desirable if used in multigrid context, where we aim to obtain optimal $\mathcal{O}(n)$ computational complexity method. In order to overcome the super-linear computational complexity of the algorithms for exact weighted matching, we tested an algorithm which allows to obtain a matching in a general graph with weight about 1/2 of the maximum, also known as half-approximate matching, with $\mathcal{O}(n)$ computational complexity [19].

Motivated by the wide range of applications, to obtain linear-time approximate algorithms with increasing performance ratio as well as effective parallel implementations of such approximate algorithms is currently an active area of research (see for example, [4, 12,16]) Our aim here is to assess the impact of using half-approximate matching in Algorithm 1 during the coarsening process described in Section 3 as well as its impact on the convergence behavior and setup cost of our composite αAMG. All results discussed in what follows are obtained using our Matlab implementation of the adaptive AMG where the `HSL-MC64` subroutine was substituted by a Matlab function implementing the matching algorithm described in [19]. We report tables that contain the parameters describing the setup cost of the composite AMG for the same test cases introduced in Section 5 for both unstructured and structured meshes. Algorithmic choices and parameter settings are the same as before, however now instead of using $W(1, 1)$ cycle, in order to have optimal multigrid components coupled with linear complexity matching, we apply a hybrid V-W cycle which allows to obtain a $\mathcal{O}(n)$ linear complexity cycle. Again, we use one sweep of symmetric Gauss-Seidel both as pre/post smoother and coarsest level solver.

5.2.1 Unstructured mesh

As in the case of exact bipartite matching, we first discuss results obtained on triangular meshes with a total

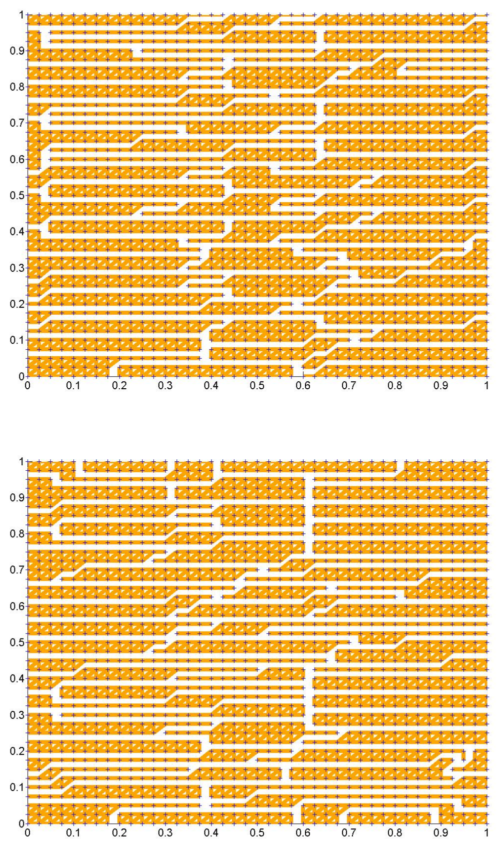

Fig. 5.9 *Test Case 4 ($\theta = \pi/3$)*, $n = 40 \times 40$. Coarsest interpolation matrices pattern built by Algorithm 2 (top) and Algorithm 3 (bottom) with exact bipartite matching.

number of nodes $n = 2705, 10657,\ 42305$, respectively, corresponding to three different mesh sizes and all five test cases (with angle of anisotropy $\theta = 0,\ \pi/8,\ \pi/3,\ \pi/4,\ \pi/2$). We report the results in Tables 5.9–5.13 the characteristics of the setup cost of the composite AMG needed to achieve pre-selected convergence factor $\rho_{desired} = 0.7$.

Composite αAMG Setup, coarsening based on Algorithm 2

n	*nstages*	ρ	*nlev*	*cmpx*
2705	3	0.49	7	2.13
10675	4	0.62	9	2.16
42305	6	0.67	12	2.17

Composite αAMG Setup, coarsening based on Algorithm 3

n	*nstages*	ρ	*nlev*	*cmpx*
2705	3	0.54	7	2.12
10657	5	0.59	9	2.16
42305	6	0.68	12	2.17

Table 5.9 *Test Case 1: ($\theta = 0$).* Setup cost for different mesh sizes when approximate graph matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2

n	*nstages*	ρ	*nlev*	*cmpx*
2705	1	0.47	7	2.12
10675	1	0.63	9	2.15
42305	2	0.54	11	2.16

Composite αAMG Setup, coarsening based on Algorithm 3

n	*nstages*	ρ	*nlev*	*cmpx*
2705	1	0.49	7	2.14
10657	1	0.64	9	2.17
42305	2	0.56	11	2.17

Table 5.10 *Test Case 2: ($\theta = \pi/8$)* Setup cost for different mesh sizes when approximate graph matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2

n	*nstages*	ρ	*nlev*	*cmpx*
2705	3	0.59	7	2.12
10675	5	0.58	9	2.15
42305	7	0.65	12	2.16

Composite αAMG Setup, coarsening based on Algorithm 3

n	*nstages*	ρ	*nlev*	*cmpx*
2705	3	0.54	7	2.12
10657	4	0.69	9	2.16
42305	8	0.70	12	2.17

Table 5.11 *Test Case 3: ($\theta = \pi/4$)* Setup cost for different mesh sizes when approximate graph matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2

n	*nstages*	ρ	*nlev*	*cmpx*
2705	3	0.46	7	2.12
10675	4	0.65	9	2.16
42305	6	0.70	12	2.17

Composite αAMG Setup, coarsening based on Algorithm 3

n	*nstages*	ρ	*nlev*	*cmpx*
2705	3	0.45	7	2.13
10657	5	0.57	9	2.16
42305	7	0.70	12	2.17

Table 5.12 *Test Case 4: ($\theta = \pi/3$)* Setup cost for different mesh sizes when approximate graph matching is used for aggregation.

A general observation is that as in the case of exact bipartite matching, the two coarsening algorithms give similar convergence behavior also in the case of approximate matching. Indeed, looking at the coarsening trees obtained per each AMG component (see Figures 5.10 and 5.11) we observe the same behavior showed in

Composite αAMG Setup, coarsening based on Algorithm 2				
n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	3	0.55	7	2.12
10675	4	0.57	9	2.16
42305	6	0.67	12	2.17

Composite αAMG Setup, coarsening based on Algorithm 3				
n	*nstages*	*ρ*	*nlev*	*cmpx*
2705	2	0.67	7	2.12
10657	4	0.62	9	2.16
42305	6	0.68	12	2.17

Table 5.13 *Test Case 5: ($\theta = \pi/2$)* Setup cost for different mesh sizes when approximate graph matching is used for aggregation.

Section 5.1. On the other hand, using half-approximate

$$\begin{tikzcd}
 & & & & & & A^1 \arrow[dl, "\varrho_c=0.98"'] \arrow[dr, "\varrho_f=0.83"] & \\
 & & & & & A_c^1 = A^2 \arrow[dl, "\varrho_c=0.98"'] \arrow[dr, "\varrho_f=0.80"] & & A_f^1 \\
 & & & & A_c^2 = A^3 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.80"] & & A_f^2 & \\
 & & & A_c^3 = A^4 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.73"] & & A_f^3 & & \\
 & & A_c^4 = A^5 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.74"] & & A_f^4 & & & \\
 & A_c^5 = A^6 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.69"] & & A_f^5 & & & & \\
A_c^6 = A^7 & & A_f^6 & & & & &
\end{tikzcd}$$

Fig. 5.10 *Test Case 5 ($\theta = \pi/2$), n =* 2705. Coarsening tree based on Algorithm 2 and approximate graph matching.

matching produces coarsening with factor less than two due to a larger number of singletons than in the case of aggregation based on exact matching. This happens because, while the exact weighted matching implemented in `HSL-MC64` computes weighted matching with maximum cardinality (perfect matching for non-singular matrices), the approximate algorithm from [19] computes a maximal weighted matching, not necessarily of maximum cardinality. The reduced coarsening factor affects the number of coarsening levels (on average by one), and as a result leads to a slight increase of the average operator complexity. We recall, that we stop the coarsening process when the size of the coarse problem reaches certain threshold. On the other hand, as expected, the use of the hybrid V-W cycle generally affects the scalability of the composite AMG. Indeed in all test cases, we observe a slight increase in the number

$$\begin{tikzcd}
 & & & & & & A^1 \arrow[dl, "\varrho_c=0.98"'] \arrow[dr, "\varrho_f=0.82"] & \\
 & & & & & A_c^1 = A^2 \arrow[dl, "\varrho_c=0.98"'] \arrow[dr, "\varrho_f=0.84"] & & A_f^1 \\
 & & & & A_c^2 = A^3 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.79"] & & A_f^2 & \\
 & & & A_c^3 = A^4 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.77"] & & A_f^3 & & \\
 & & A_c^4 = A^5 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.75"] & & A_f^4 & & & \\
 & A_c^5 = A^6 \arrow[dl, "\varrho_c=0.99"'] \arrow[dr, "\varrho_f=0.77"] & & A_f^5 & & & & \\
A_c^6 = A^7 & & A_f^6 & & & & &
\end{tikzcd}$$

Fig. 5.11 *Test Case 5 ($\theta = \pi/2$), n =* 2705. Coarsening tree based on Algorithm 3 and approximate graph matching.

of components (1 or 2 additional components, except for *Test Case 3* with the largest mesh and Algorithm 3, which requires 3 additional components) needed to reach the desired convergence factor. In Figures 5.12–5.13 we show the pattern of the aggregates built by the two coarsening algorithms for the first component of the composite AMG solvers built for *Test Case 1* and *Test Case 3* with the smallest mesh. In these cases the number of aggregates at the coarsest level is $n_c = 72$ (top) and $n_c = 69$ (bottom) in Figure 5.12, $n_c = 74$ (top) and $n_c = 67$ (bottom) in Figure 5.13. The figures in both cases show that both coarsening algorithms are able to fairly detect the direction of anisotropy however not as accurately as the aggregates obtained using exact matching (see Figures 5.3 and 5.4). The latter appear better aligned with the direction of anisotropy than the ones obtained using half-approximate matching.

5.2.2 Structured mesh

In the following we report the same results as in the previous subsection corresponding to the construction of composite AMG with a desired convergence rate, now on structured meshes, using half-approximate matching for aggregation. Also, in order to have optimal $\mathcal{O}(n)$ setup complexity for each solver component, we apply the hybrid V-W cycle (to compensate for the coarsening factor less than two). All other algorithmic choices are the same as in Section 5.1.2.

We observe from Tables 5.14–5.16, that using half-approximate matching coupled with the hybrid V-W does not affect in a significant way the convergence behavior of the constructed composite αAMG. We generally see, and not in all cases, only an increase of one

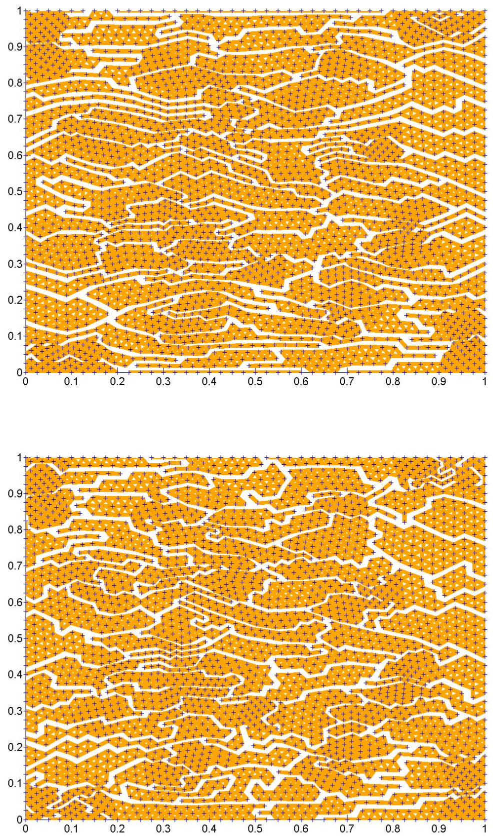

Fig. 5.12 *Test case 1 ($\theta = 0$), $n = 2705$.* Coarsest interpolation matrices pattern built by Algorithm 2 (top) and Algorithm 3 (bottom) with approximate graph matching.

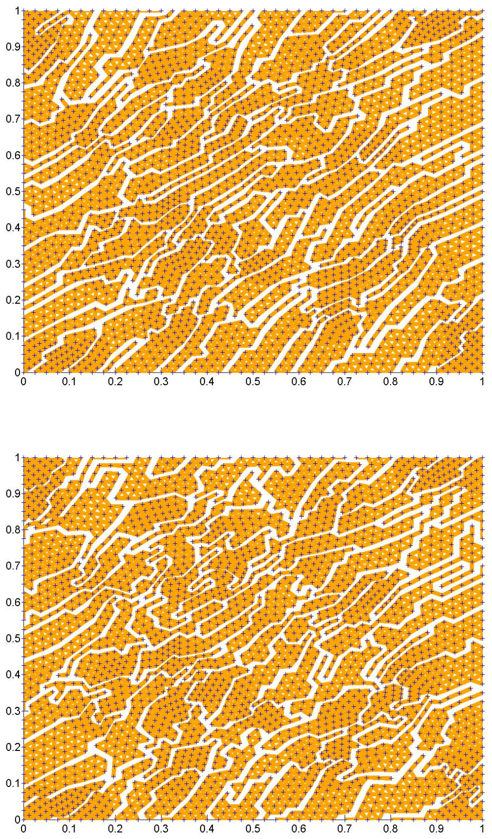

Fig. 5.13 *Test case 3 ($\theta = \pi/4$), $n = 2705$.* Coarsest interpolation matrices pattern built by Algorithm 2 and Algorithm 3 (bottom) with approximate graph matching.

Composite αAMG Setup, coarsening based on Algorithm 2				
n	*nstages*	*ρ*	*nlev*	*cmpx*
64 × 64	1	0.62	7	2.12
128 × 128	2	0.59	10	2.17
256 × 256	3	0.66	12	2.19

Composite αAMG Setup, coarsening based on Algorithm 3				
n	*nstages*	*ρ*	*nlev*	*cmpx*
64 × 64	2	0.59	8	2.23
128 × 128	3	0.67	10	2.30
256 × 256	5	0.63	12	2.36

Table 5.14 *Test Case 1: ($\theta = 0$).* Setup cost for different mesh sizes when approximate graph matching is used for aggregation.

Composite αAMG Setup, coarsening based on Algorithm 2				
n	*nstages*	*ρ*	*nlev*	*cmpx*
64 × 64	1	0.58	8	2.00
128 × 128	2	0.47	10	2.02
256 × 256	3	0.62	12	2.04

Composite αAMG Setup, coarsening based on Algorithm 3				
n	*nstages*	*ρ*	*nlev*	*cmpx*
64 × 64	2	0.60	8	2.00
128 × 128	2	0.57	10	2.05
256 × 256	3	0.62	12	2.07

Table 5.15 *Test case 3: ($\theta = \pi/4$).* Setup cost for different mesh sizes when approximate graph matching is used for aggregation.

additional solver component versus the counterpart results discussed in Section 5.1.2.

In Figures 5.14–5.16, we show the pattern of the interpolation matrices built by our two coarsening processes for 40 × 40 rectangular fine mesh.

The last figures also show that using half-approximate matching produces aggregates with fairly similar quality compared with the aggregates obtained by the exact matching displayed in Figs. 5.7-5.8-5.9.

6 Concluding Remarks

In this paper we have performed a preliminary study about two coarsening algorithms based on exact and

Composite αAMG Setup, coarsening based on Algorithm 2				
n	*nstages*	ρ	*nlev*	*cmpx*
64 × 64	3	0.57	8	2.04
128 × 128	4	0.61	10	2.06
256 × 256	8	0.62	12	2.07

Composite αAMG Setup, coarsening based on Algorithm 3				
n	*nstages*	ρ	*nlev*	*cmpx*
64 × 64	3	0.62	8	2.08
128 × 128	5	0.56	10	2.11
256 × 256	8	0.65	12	2.12

Table 5.16 *Test case 4: ($\theta = \pi/3$).* Setup cost for different mesh sizes when approximate graph matching is used for aggregation.

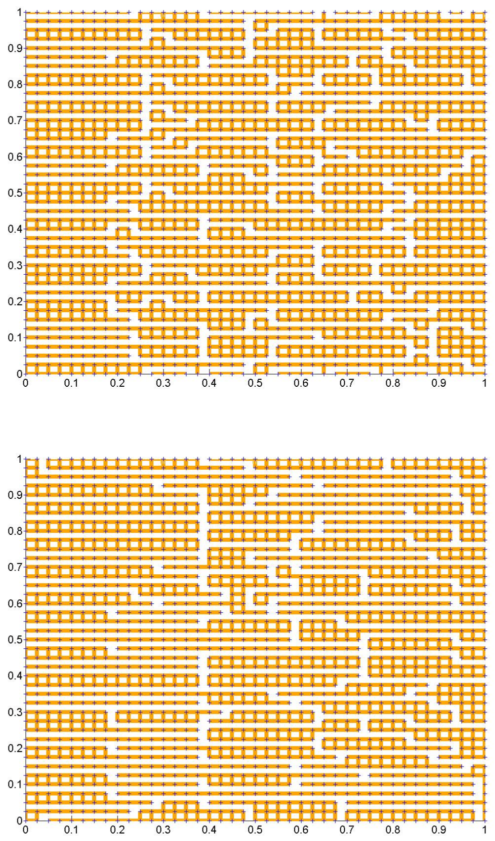

Fig. 5.14 *Test Case 1 ($\theta = 0$)*, $n = 40 \times 40$. Coarsest interpolation matrices pattern built by Algorithm 2 (top) and Algorithm 3 (bottom) with approximate graph matching.

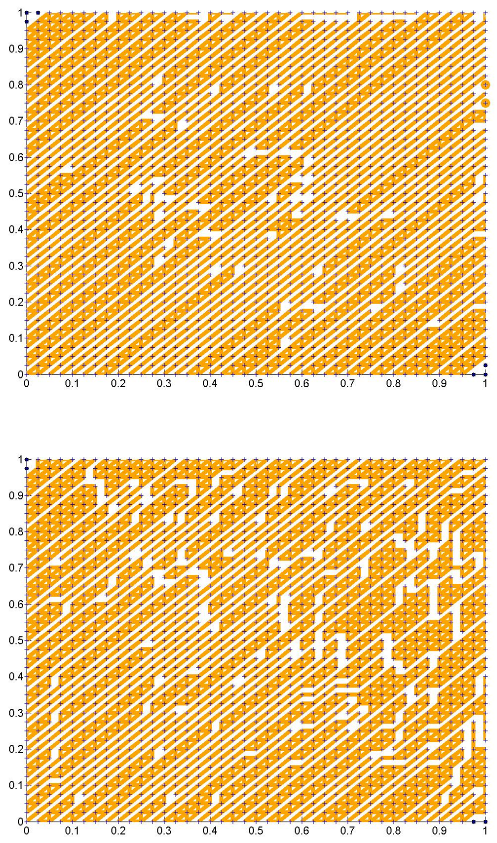

Fig. 5.15 *Test Case 3 ($\theta = \pi/4$)*, $n = 40 \times 40$. Coarsest interpolation matrices pattern built by Algorithm 2 (top) and Algorithm 3 (bottom) with approximate graph matching.

approximate maximum product matching in a graph employed in a new composite adaptive AMG process. By performing a large set of experiments on difficult for AMG non-grid aligned finite element 2nd order anisotropic elliptic equations, we demonstrated that our approach can lead to semi-coarsening and to overall composite αAMG solver with desired pre-set convergence factor. The composite solver can become expensive since the number of components that are built generally increase when refining the mesh. This is perhaps to be expected for AMG solvers for such non-grid aligned anisotropic problems when standard (pointwise) smoothers are employed. In such cases one way to alleviate the cost might be to combine several components in one cycle, by using larger aggregates and several algebraically smooth vectors to build one tentative interpolation matrix.

Another direction to exploit in the future is to test the method on systems of PDEs (like elasticity) and more general sparse matrices not necessarily coming from PDEs. Finally, parallel versions of (approximate) matching algorithms can be exploited to construct AMG solvers suitable for large-scale computations.

Acknowledgements We wish to thank Bora Uçar for stimulating discussions and advice on available algorithms and software for computing weighted matching in bipartite and general graphs.

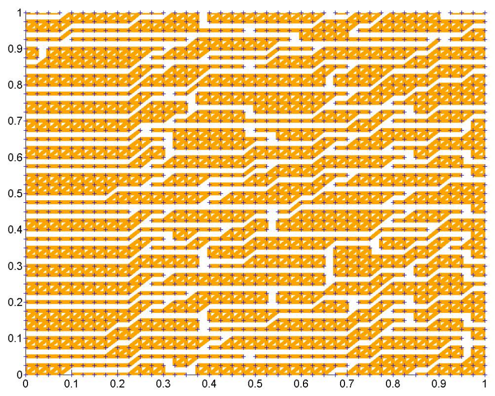
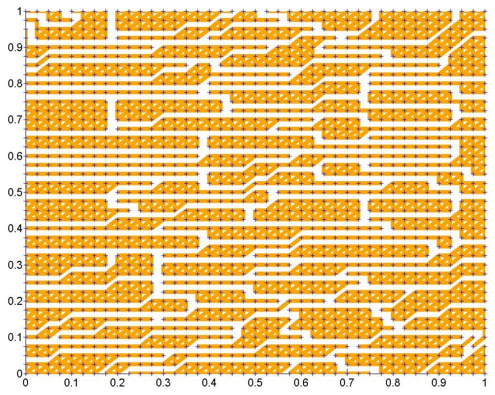

Fig. 5.16 *Test Case 4 ($\theta = \pi/3$),* $n = 40 \times 40$. Coarsest interpolation matrices pattern built by Algorithm 2 and Algorithm 3 (bottom) with approximate graph matching.

References

1. A. Brandt, *General Highly Accurate Algebraic Coarsening*, Electronic Transactions on Numerical Analysis, Vol. 10, pp. 1–20 (2000).
2. A. Brandt, *Multiscale Scientific Computation: Review* 2001, in Multiscale and Multiresolution Methods: Theory and Applications, T. J. Barth, T. F. Chan, and R. Haimes, eds., Springer, Heidelberg, pp. 1–96 (2001).
3. A. Brandt, J. Brannick, K. Kahl, I. Livshitz, *Bootstrap AMG*, Siam Journal on Scientific Computing, Vol. 33, pp. 612–632 (2011).
4. D. E. Drake, S. Hougardy, *A Linear Time Approximation Algorithm for Weighted Matchings in Graphs*, ACM Transactions on Algorithms, Vol. 11, pp. 107–122 (2005).
5. A. Brandt, S. McCormick, J. Ruge, *Algebraic Multigrid (AMG) for Sparse Matrix Equations.* In Sparsity and its Applications, D. J. Evans ed. (1984).
6. J. Brannick, R. D. Falgout, *Compatible Relaxation and Coarsening in Algebraic Multigrid*, SIAM Journal on Scientific Computing, Vol. 32, pp. 1393–1416 (2010).
7. M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, J. Ruge, *Adaptive Smoothed Aggregation αSA Multigrid*, SIAM Review, Vol. 47, pp. 317–346 (2005).
8. M. Brezina, R. Falgout, S. MacLachlan, T. Manteuffel, S. McCormick, J. Ruge, *Adaptive Algebraic Multigrid*, SIAM Journal on Scientific Computing, Vol. 27, pp. 1261–1286 (2006).
9. I. S. Duff, J. Koster, *On Algorithms for Permuting Large Entries to the Diagonal of a Sparse Matrix*, Rutherford Appleton Laboratory Technical Report RAL-TR-1999-030.
10. I. S. Duff, B. Uçar, *Combinatorial Problems in Solving Linear Systems*, Rutherford Appleton Laboratory Technical Report RAL-TR-2008-014 and TR/PA/08/26, CERFACS, Toulouse.
11. R. Falgout, P. S. Vassilevski, *On Generalizing the AMG Framework*, SIAM Journal on Numerical Analysis Vol. 42, pp. 1669–1693 (2004).
12. M. Halappanavar, J. Feo, O. Villa, A. Tumeo, A. Pothen, *Approximate Weighted Matching on Emerging Manycore and Multithreaded Architectures*, Int. J. of High-Performance Computing Applications, first published on August 9, 2012 as doi:10.1177/1094342012452893.
13. HSL(2011). A Collection of Fortran Codes for Large Scale Scientific Computation. http://www.hsl.rl.ac.uk.
14. I. Lashuk, P. S. Vassilevski, *On Some Versions of the Element Agglomeration AMGe Method*, Numerical Linear Algebra with Applications Vol. 15, pp. 595–620 (2008).
15. O. E. Livne, *Coarsening by Compatible Relaxation*, Numerical Linear Algebra with Applications, Vol. 11, pp. 205–227 (2004).
16. F. Manne, R. H. Bisseling, *A Parallel Approximation Algorithm for the Weighted Maximum Matching Problem*, in PPAM 2007, LNCS, Vol. 4967, Springer-Verlag, pp. 708–717 (2008).
17. M. Olschowka, A. Neumaier, *A New Pivoting Strategy for Gaussian Elimination*, Linear Algebra and its Applications, Vol. 240, pp. 131–151 (1996).
18. S. Parter, *The Use of Linear Graphs in Gaussian Elimination,* SIAM Review Vol. 3, pp. 119-130 (1961).
19. R. Preis, *Linear Time 1/2-Approximation Algorithm for Maximum Weighted Matching in General Graphs*, in STACS'99, LNCS, Springer-Verlag, Vol. 1563, pp. 259-269 (1999).
20. P. Vaněk, J. Mandel, and M. Brezina, *Algebraic Multigrid by Smoothed Aggregation for Second and Fourth Order Elliptic Problems*, Computing, Vol. 56, pp. 179–196 (1996).
21. P. S. Vassilevski, *Multilevel Block Factorization Preconditioners, Matrix-based Analysis and Algorithms for Solving Finite Element Equations*, Springer: New York, NY (2008).