EVALUATION OF PARALLEL AGGREGATE CREATION ORDERS: SMOOTHED AGGREGATION ALGEBRAIC MULTIGRID METHOD

Akihiro Fujii
The Center for Continuing Professional Development, Kogakuin University 1-24-2, Nishishinjuku, Shinjuku-ku, Tokyo, Japan; CREST, JST, 4-1-8 Honcho Kawaguchi, Saitama, Japan.
fujii@cpd.kogakuin.ac.jp

Akira Nishida
Department of Computer Science, Graduate School of Information Science and Technology, University of Tokyo 7-3-1, Hongo, Bunkyo-ku, Tokyo, Japan; CREST, JST, 4-1-8 Honcho Kawaguchi, Saitama, Japan.
nishida@is.s.u-tokyo.ac.jp

Yoshio Oyanagi
Department of Computer Science, Graduate School of Information Science and Technology, University of Tokyo 7-3-1, Hongo, Bunkyo-ku, Tokyo, Japan
oyanagi@is.s.u-tokyo.ac.jp

Abstract The Algebraic MultiGrid method (AMG) has been studied intensively as an ideal solver for large scale Poisson problems. The Smoothed Aggregation Algebraic MultiGrid (SA-AMG) method is one of the most efficient of these methods. The aggregation procedure is the most important part of the method and is the main area of interest of several researchers.

Here we investigate aggregate creation orders in the aggregation procedure. Five types of aggregation procedure are tested for isotropic, anisotropic and simple elastic problems. As a result, it is important that aggregates are created around one aggregate in each domain for isotropic problems. For anisotropic problems, aggregates around domain borders influence the convergence much. The best strategy for both anisotropic and isotropic problems in our numerical test is the aggregate creation method which creates aggregates on borders first then creates aggregates around one aggregate in the internal domain.

In our test, the SA-AMG preconditioned Conjugate Gradient (CG) method is compared to the Localized ILU preconditioned CG method. In the experiments, Poisson problems up to 1.6×10^7 DOF are solved on 125PEs.

Keywords: AMG; Poisson Solver; Aggregate Creation

1. Introduction

For large-scale Poisson problems, multi-level methods are known to be efficient solvers. In multi-level methods a smaller problem is constructed from the problem matrix in the setup phase, and this is used to solve the problem in the solution phase. The Smoothed Aggregation Algebraic MultiGrid (SA-AMG) method is one of the most effective multi-level methods. In the SA-AMG method the smaller problem matrices are calculated by creating aggregates of the neighboring unknowns. The convergence efficiency depends on the quality of the aggregates.

The parallel SA-AMG method is often realized by domain decomposition, and parallel aggregation strategies based on domain borders are needed. Thus, parallel aggregation strategies are a very important part of the method, and are discussed in [Ada98, TT00]. These papers propose and compare various aggregation strategies, but they do not consider the order in which the aggregates are created. The order of the aggregate creation also highly influences the convergence. We implement the parallel SA-AMG method which can deal with any aggregation strategy, and investigate which order of aggregate creation is better for both isotropic and anisotropic problems.

Section 2 introduces the SA-AMG method. Section 3 explains the various orders of aggregate creation, and Section 4 summarizes the implementation of the SA-AMG method, followed by numerical experiments and conclusions.

2. SA-AMG Method

The SA-AMG method is one of the MultiGrid methods. First, the MultiGrid method is explained.

It is difficult to solve large-scale problems but the MultiGrid method utilizes more than one grid and solves large-sized problems efficiently. For example, we consider the two grid case of fine and coarse grids. The problem matrices A_1 and A_2 are discretized on the fine grid and the coarse grid respectively. Thus, matrix A_1 is bigger than matrix A_2.

The problem is to solve the equation $A_1x_1 = b_1$, where vector x_1 is the unknown vector. The MultiGrid solution process is as follows.

1 Perform a relaxation, such as the Gauss Seidel method, for the equation $A_1x_1 = b_1$ on the fine grid

2 Compute the residual $r_1 = b_1 - A_1x_1$, and move r_1 to the coarse grid $r_2 = Rr_1$

3 Solve the coarse grid residual equation $A_2x_2 = b_2$ (possibly recursively)

4 Move x_2 to the fine grid and correct the fine grid solution $x_1 \leftarrow x_1 + Px_2$

5 If the convergence criterion isn't satisfied, go to 1

Matrix R represents the restriction operator which moves the vector on the fine grid to the vector on the coarse grid. Matrix P represents the prolongation operator which moves the vector on the coarse grid to the vector on the fine grid. Matrix R is often the transpose of matrix P. Although we consider here a two-level case, Step 3 can be executed recursively and more than two grid cases can be extended to multi-level cases. The MultiGrid method is known to be a very effective method for problems with structured meshes but it is difficult to create coarse grids for unstructured problems. Algebraic MultiGrid (AMG) methods have been studied intensively for such problems. AMG methods create the smaller problem from only the problem matrix without mesh information.

AMG methods have the two phases of setup and solution. The solution phase solves the problem as described above. The setup phase of the AMG method is explained here. The setup phase creates a smaller matrix from the problem matrix. AMG methods often calculate the prolongation matrix P and $R = P^T$ first. Then the smaller problem matrix is calculated from $A_2 = RA_1P$, which is called the Galerkin approximation. If more than one coarse problem matrix is utilized, this process is repeated and the coarser level matrices are calculated in order.

Various AMG methods differ in how the prolongation matrix P is created from the problem matrix A_1. In the SA-AMG method the prolongation matrix P is calculated using the following three procedures.

Filtering the Matrix A_1 Matrix $\tilde{A}_1$ is defined from matrix A_1 by dropping the elements that do not satisfy the following condition.

$$a_{ij}^2 > \theta^2 |a_{ii}||a_{jj}| \tag{1}$$

where θ is a constant value between 1 and 0. $a_{i,j}$ means the element of the i-th row and j-th column of matrix A_1.

Construction of Aggregates We define a graph for the symmetric matrix $\tilde{A}_1$. The vertices and edges in the graph correspond to rows and non-zero elements of matrix $\tilde{A}_1$. The vertex set is divided into disjoint subsets containing neighboring vertices following this procedure.

phase1 : repeat until all unaggregated vertices are adjacent to an aggregate

- pick a root vertex which is not adjacent to any existing aggregate
- define a new aggregate as the root vertex plus all its neighbors

phase2 : sweep unaggregated vertices into existing aggregates or use them to form new aggregates

Figure1 shows the situation. Each subset is called an aggregate, and corresponds to an unknown of the next coarsest level.

Matrix $\tilde{P}$ is an $n \times m$ matrix, where n and m are the numbers of unknowns and aggregates respectively, and is defined as follows:

$$\tilde{P}_{ij} = \begin{cases} 1 & \text{the i-th unknown belongs to the j-th aggregate} \\ 0 & \text{other cases} \end{cases}$$

Relaxation of Aggregates The SA-AMG method defines the prolongation matrix by smoothing the matrix $\tilde{P}$ defined above. Here we assume that the damped Jacobi method is used for smoothing.

$$P = (I - \omega \tilde{D}^{-1} \tilde{A}_1)\tilde{P} \tag{2}$$

where ω and $\tilde{D}$ are the damping coefficient and the diagonal part of $\tilde{A}_1$, respectively. Other methods of smoothing aggregates are described in [VBM01].

Convergence of the SA-AMG method depends significantly on the quality of the multi-level data structure constructed in the setup phase. The construction of aggregates is especially important. For fast convergence, aggregates need to be packed tightly in the domain. As in Figure 2, adjacent aggregates are often created in order. In a parallel computing environment, the problem domain is often decomposed and

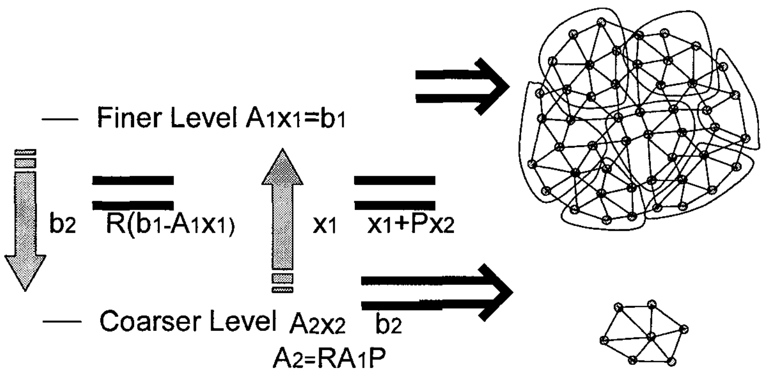


Figure 1. Conceptual Image of the SA-AMG method: Graph structure is obtained from the matrix

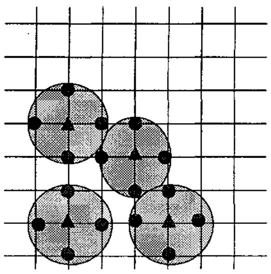


Figure 2. Aggregation Phase1 for a Five-Point Stencil: The triangle identifies the root vertex and the small circle identifies a vertex neighboring the root vertex

the aggregation process is executed by each PE. In the next section we discuss strategies for aggregate creation orders for parallel aggregation.

3. Various Orders of Aggregate Creation

Parallel aggregation strategies are discussed in this section. These are classified into two types, independent and joint. Independent aggregation means that the aggregation is carried out independently in each domain. Joint aggregation first makes shared aggregates around borders and then creates aggregates independently in the inner domain.

Adams [Ada98] proposed the Maximal independent set algorithm as an aggregation strategy. It has become one of the joint aggregation strategies. Tuminaro et al. [TT00] proposed another joint aggregation strategy and compared various aggregation strategies. They discussed a method for realizing joint aggregation, but did not investigate aggregate

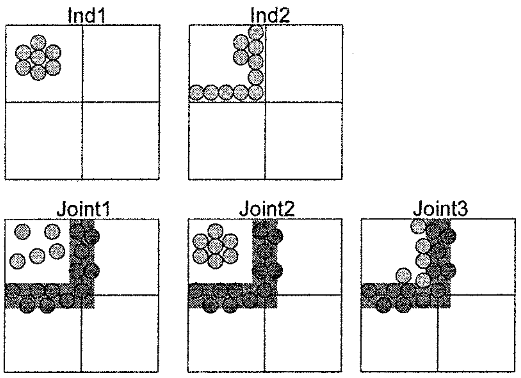


Figure 3. Two Types of Independent Aggregation: Ind1 and Ind2. Three Types of Joint Aggregation: Joint1, Joint2 and Joint3. The range colored orange represents the vertices on borders with neighboring PEs. A circle represents an aggregate. A green circle represents an aggregate whose root vertex is on a border, and a blue circle represents an aggregate whose root vertex is an internal vertex of the domain.

creation orders. We investigate the various aggregate creation orders for better convergence.

This paper considers and compares five types of aggregate-creation method as follows. Figure 3 shows the situation.

- Independent Aggregation: Aggregates are created independently from other PEs and aggregates cannot go beyond the borders.

  - Ind1: Aggregates are created around one aggregate in order.
  - Ind2: Aggregates are created from borders in order.

- Joint Aggregation: Root vertices are selected in vertices of borders, greedily at first. Then, root vertices are selected in the interior domain by each PE. The following three strategies differ in the method used to create aggregates in the interior domain.

  - Joint1: Aggregates in the interior domain are created greedily.
  - Joint2: Aggregates in the interior domain are created around one aggregate in order similar to Ind1.
  - Joint3: Aggregates in the interior domain are created around aggregates on borders.

4. Implementation

Our implementation of the SA-AMG method is based on Tuminaro et al. [TT00] and GeoFEM [Geo]. The joint aggregations method is based on Tuminaro et al. [TT00]. Data structure for finite element problems, ICCG solvers and other parts of the implementation are based on GeoFEM.

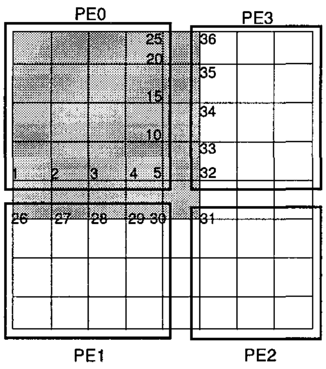
This section explains the implementation of our solver. The data structure of matrices and the resulting form of the aggregates, which are an important part of the implementation, are discussed.

Firstly, the data structure of the matrices is written. The graph based on the problem matrix is decomposed into sub-domains for PEs. Then, rows and columns of the matrix are reordered by the sub-domains, and the reordered matrix is distributed as a one-dimensional block-row distribution. Each PE has a block of rows from the problem matrix.

Vertices in each PE's domain are also connected with the vertices, called ghost vertices, in the neighboring domain. Each PE's calculation requires the values of the ghost vertices. Thus, communication tables which record the PE number and the vertex number of the ghost vertices are also necessary. Figure 4 shows a simple 2-dimensional finite-element mesh with four PEs. Vertices of the gray part of Figure 4 are required for calculating PE 0's domain, and the vertices 1..25. The ghost vertices are 26..36. $neibPE(:)$, $send(:,p)$ and $recv(:,p)$ have neighboring PE numbers, their own vertices referred to by PE $neibPE(p)$ and the vertex number of the ghost vertices owned by PE $neibPE(p)$.

Secondly, the resulting form of the aggregation is written. Parallel aggregation creates aggregates which are disjoint sets of vertices on the graph based on the problem matrix of each level. It then determines the owner PE for each aggregate. PE k's own aggregates are collected to PE k as vertices on the next coarser level. Thus, the coarser level's domain decomposition follows the determination of the owner PE of the aggregates. The relationship between the owner PE and the aggregates can be any combination, and the domain decomposition of the finer and the coarser levels can be totally changed.

The information for the aggregates is recorded by each PE. There are two types of aggregate recorded by a PE, its own aggregates and external aggregates. A PE's own aggregates are owned by the PE on the next coarsest level. External aggregates are owned by the other PE. Figure 5 shows PE k's 2-dimensional array which contains the vertex numbers of the aggregates. In Figure 5, PE k recorded N_a aggregates. Aggregate numbers $1..N_c$ are the aggregates owned by PE k on the next coarsest level. Aggregate numbers $N_c + 1..N_a$ are the aggregates owned



$$\begin{aligned}
neibPE(:) &= 1,2,3\\
send(:,1) &= 1,2,3,4,5\\
send(:,2) &= 5\\
send(:,3) &= 5,10,15,20,25\\
recv(:,1) &= 26,27,28,29,30\\
recv(:,2) &= 31\\
recv(:,3) &= 32,33,34,35,36
\end{aligned}$$

Figure 4. Vertices and Communication Tables of PE 0

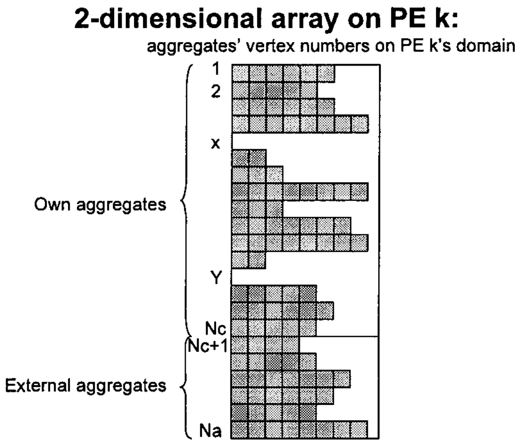

2-dimensional array on PE k:
aggregates' vertex numbers on PE k's domain
1
2
X
Own aggregates
Y
Nc
Nc+1
External aggregates
Na

Figure 5. PE k's Aggregate Information Format: Aggregates $1.N_c$ are owned by PE k on the coarser level. Vertex numbers on PE k's domain are recorded in the array. Aggregates X and Y are owned by PE k on the coarser level, but aggregates X and Y are not on the vertices on PE k's domain.

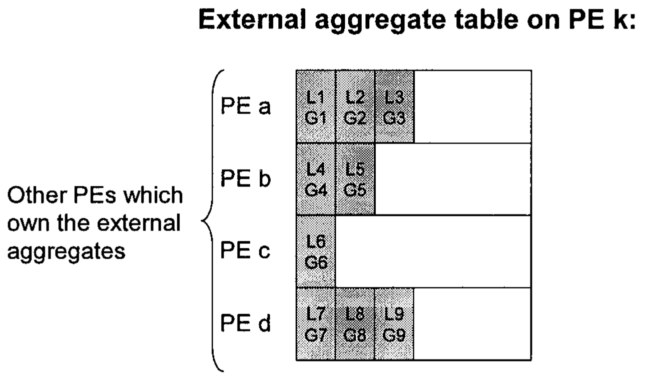

External aggregate table on PE k:
Other PEs which own the external aggregates
PE a
PE b
PE c
PE d
L1 G1
L2 G2
L3 G3
L4 G4
L5 G5
L6 G6
L7 G7
L8 G8
L9 G9

Figure 6. PE k's Aggregate Table: Aggregate Table records pairs of the global aggregate number (G?) and the local aggregate number (L?) for the external aggregates.

by other PEs. The array has the aggregates' vertex numbers in PE k's domain. Thus, one aggregate, which contains vertices on the domain of many PEs, is recorded by many PEs. Figure 5 shows that aggregates X and Y have no vertices in PE k's domain and no vertices are recorded. But the two aggregates may have vertices in the other PE's domains and may be recorded as an external aggregate by those PEs. Aggregates are recorded by PE k in the following cases.

- The aggregate's owner PE is PE k.
- The aggregate's vertices contain any vertices of PE k's domain.

Some aggregates are recorded by many PEs, and a global aggregate number is needed to identify them. The local aggregate number is determined and utilized as in Figure 5. The PEs own aggregates are numbered from 1 to the number of owned aggregates (N_c in Figure 5) for each PE. Then, external aggregates follow. In our implementation the global aggregate number is determined by numbering the owned aggregates in the order of PE numbers. For owned aggregates on PE k, the global aggregate number is determined by adding the local aggregate number to the sum of the number of aggregates owned by PEs 0..k-1. External aggregates, however, cannot be identified by a local aggregate number. Therefore, their global aggregate numbers are recorded in an aggregate table as shown in Figure 6. The external aggregate table records the pairs of global and local aggregate numbers for external aggregates.

Ultimately, the output of the aggregation on each PE is recorded as two arrays, sets of vertices (Figure 5) and an aggregate table (Figure 6).

This output interface can deal with any vertex set as an aggregate and can cope with any PE allocation for aggregates.

We assume that the aggregation procedure determines two things: the vertex set of each aggregate and the owner PE of each aggregate on the next coarsest level. The implementation can deal with any aggregation strategy which follows such assumptions.

5. Numerical Experiments and Evaluations

In this section, we evaluate the parallel algorithm for three types of problem: anisotropic, isotropic, and 3-dimensional elastic. The problem size per PE is set to be constant for each type of problem in order to understand the behavior of the solver for large-size problems on massively parallel systems.

The SA-AMG mathod and Localized ILU preconditioned CG method [NO99, NNT97, Nak03], which is refered to as ICCG, are compared in the experiments. In addition, five aggregation strategies for the SA-AMG method are tested. These are introduced in section 3.

The SA-AMG method utilizes a V-cycle for the solution phase. One iteration of the Chaotic Symmetric Gauss Seidel(Chaotic SGS) method is carried out as pre- and post-smoothing at each level. At the coarsest level, twenty iterations of the Chaotic SGS method or parallel direct solution method are performed. We utilize PSPASES [GGJ$^+$97] as the parallel direct solver.

Numerical Experiment Environment Numerical experiments are carried out on a cluster with 128 nodes of Sun Blade1000 workstations which have dual CPUs of UltraSPARC III 750MHz and 1GB memory. These nodes are connected by Myrinet2000. For parallelization, MPICH-GM [MPI, Myr] is used. The code is written in Fortran90.

5.1 Anisotropic Problems

Equation 3 with a rectangular parallelepiped domain discretized by a finite element method is solved for anisotropic and isotropic problems. The problem domain has six surfaces perpendicular to the x, y and z axes. In Equation 3 Xmin, Ymin, Zmax and Zmin represent surfaces with minimum X value, minimum Y value, maximum Z value and minimum Z value respectively. In the anisotropic case, ϵ in equation 3 is set to 0.0001. This anisotropic problem is very difficult to solve, from the view point of condition number.

Table 1. Problem Size and Number of PEs for Anisotropic Problems: DOF represents Degree of Freedom of the problem.

Anisotropic Problems		
DOF	# of PEs	Problem Domain
8192k	128	160 × 160 × 320
4096k	64	160 × 160 × 160
2048k	32	80 × 160 × 160
1024k	16	80 × 80 × 160
512k	8	80 × 80 × 80
256k	4	40 × 80 × 80
128k	2	40 × 40 × 80

$$\partial/\partial x(\partial p/\partial x) + \partial/\partial y(\partial p/\partial y) + \partial/\partial z(\epsilon\, \partial p/\partial z) = 0 \tag{3}$$
$$\begin{cases} Xmin : \partial p/\partial x = 10.0 \\ Ymin : \partial p/\partial y = 5.0 \\ Zmax : \partial p/\partial z = 1.0 \\ Zmin : p = 0.0 \end{cases}$$

The problem domain is decomposed into sub-domains with $40\times40\times40$ vertices, which are allocated to each PE. Table 1 shows the size of the problems and the number of PEs. The convergence criterion is that the 2-norm of the relative residual is less than 10^{-7}.

For anisotropic problems, the coarsest level problem is still difficult to solve. We tested some aggregation strategies with a parallel direct solver at the coarsest level. There are SA-AMG methods with five levels and four levels. SA-AMG methods with five levels utilize twenty iterations of the Chaotic SGS method at the coarsest level, and SA-AMG methods with four levels utilize the parallel direct solver at the coarsest level. In Table 2, the problem size at the coarsest level is described for each aggregation strategy. The problem size is biggest in the 128PE case and that case is written.

ICCG and SA-AMG Methods Tables 3 and 4 show the total times for the ICCG and SA-AMG methods with Ind1 of independent aggregation for anisotropic problems. The ICCG method doesn't reach the convergence criterion for the biggest problem and indicates the difficulty of the problem. On the other hand, Table 4 shows that the SA-AMG method works well for anisotropic problems in comparison with

Coarsest Level Size for 128PE case		
Aggregation Strategy	Size	# of Levels
Ind1	7864	5
Ind2	6375	5
Ind2 with Parallel Direct Solver	22601	4
Joint1	2602	5
Joint2	3328	5
Joint3	3136	5
Joint1 with Parallel Direct Solver	20524	4

Table 2. Coarsest Problem Size for Anisotropic Problems

the ICCG method. The following paragraphs compare and evaluate aggregation strategies for the SA-AMG method for anisotropic problems.

Adaptation of Independent Aggregation for Anisotropy Tables 4, 5, and 6 have the iteration number, time of each phase, and the total time for the SA-AMG method with three types of independent aggregation strategy. Table 4 shows the normal independent aggregation of Ind1. This aggregation makes irregular aggregates around borders, and the iteration number for convergence increases. Table 5 shows the result of aggregation from borders, which is explained as Ind2 of independent aggregation in section 3. This improvement does well in convergence. In addition, the number of levels is reduced with the parallel direct solver at the coarsest level. The result is shown in Table 6. By using these improvements, independent aggregation can be adapted to anisotropic problems.

Joint Aggregation for Anisotropy Tables 7, 8, 9 and 10 show the results of various aggregation strategies for joint aggregation, Joint1, Joint2, Joint3, and Joint1 with fewer levels and a parallel direct solver. Joint1, Joint2 and Joint3 differ in their aggregate creation method for the inner part of each PE's domain. Joint1 creates aggregates in the order of the vertex numbers, and Joint2 creates aggregates around one aggregate in order. Joint3 creates aggregates from borders. These are further explained in section 3. Their performances for anisotropic problems differ little. It also shows that improvements in the reduction of level number and the use of a parallel direct solver, work well. In comparison with independent aggregation, joint aggregations works effectively for anisotropic problems.

ICCG method		
# of PEs	# of Iteration	Total Time
128	>10000	> 2106
64	5527	1169.7
32	5426	1100.4
16	5232	1030.2
8	2750	530.1
4	2677	506.0
2	2219	412.1

Table 3. ICCG Method for Anisotropic Problems. Time is written in seconds.

Aggregation Strategy For Anisotropic Problems For this numerical experiment, joint aggregation with a parallel direct solver is the most efficient. Ind2 of independent aggregation with a parallel direct solver, however, works nearly as well as the previous method. Except for methods using a parallel direct solver, any joint aggregation works well for anisotropic problems.

Ind1 of Independent Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
128	515	5.7	443.0	448.8
64	732	5.6	622.2	627.9
32	506	5.0	399.5	404.5
16	73	4.3	51.8	56.2
8	47	4.2	32.1	36.4
4	60	3.7	39.4	43.1
2	27	3.3	16.9	20.4

Table 4. SA-AMG Method for Anisotropic Problems: Ind1 of independent aggregation creates aggregates around one aggregate in order. The number of levels is five. Time is written in seconds.

5.2 Isotropic Problems

For isotropic problems, ϵ in equation 3 is set to 1. Problem domain, discretization method and boundary conditions are the same as the anisotropic case. The problem domain is decomposed into sub-domains with $50 \times 50 \times 50$ vertices, which are allocated to each PE. Table 11 shows the sizes of problems and the numbers of PEs. The convergence criterion is that the 2-norm of the relative residual is less than 10^{-12}.

For isotropic problems, all SA-AMG methods have four levels and utilize twenty iterations of the Chaotic SGS method at the coarsest level.

Ind2 of Independent Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
128	245	5.0	205.6	210.7
64	147	5.0	122.6	127.6
32	65	4.6	50.0	54.7
16	62	4.1	42.9	47.1
8	43	3.9	28.9	32.9
4	44	3.5	28.4	31.9
2	29	3.3	18.2	21.6

Table 5. SA-AMG Method for Anisotropic Problems: Ind2 of independent aggregation creates aggregates from borders. The number of levels is five. Time is written in seconds.

Ind2 of Independent Aggregation with fewer levels and a parallel direct solver				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
128	82	6.9	62.1	69.1
64	34	5.8	24.9	30.8
32	33	4.9	23.1	28.1
16	26	4.2	17.4	21.6
8	26	4.0	17.0	21.1
4	22	3.5	14.1	17.7
2	21	3.3	13.3	16.6

Table 6. SA-AMG Method for Anisotropic Problems: Ind2 of independent aggregation with fewer levels utilizing a parallel direct solver at the coarsest level. The number of levels is four. Time is written in seconds.

Joint1 of Joint Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
128	99	4.9	78.1	83.1
64	59	4.6	46.7	51.4
32	48	4.2	36.1	40.4
16	47	4.0	32.2	36.2
8	32	3.8	21.4	25.2
4	20	3.4	12.8	16.3
2	26	3.2	16.2	19.5

Table 7. SA-AMG Method for Anisotropic Problems: Joint1 of joint aggregation is explained in section 3. Aggregates in the internal domain are created greedily. The number of levels is five. Time is written in seconds.

Joint2 of Joint Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
128	109	5.1	89.4	94.6
64	64	5.0	52.5	57.5
32	51	4.5	37.9	42.5
16	48	4.2	33.2	37.5
8	32	4.0	22.4	26.4
4	26	3.6	16.8	20.4
2	32	3.3	20.1	23.5

Table 8. SA-AMG Method for Anisotropic Problems: Joint2 of joint aggregation is explained in section 3. Aggregates in the internal domain are created around one aggregate in order. Time is written in seconds.

Joint3 of Joint Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
128	102	5.1	83.9	89.1
64	62	4.9	50.6	55.6
32	51	4.5	38.0	42.6
16	47	4.2	32.2	36.6
8	31	4.0	20.8	24.9
4	26	3.6	16.6	20.3
2	32	3.4	20.2	23.6

Table 9. SA-AMG Method for Anisotropic Problems: Joint3 of joint aggregation is explained in section 3. Aggregates in the internal domain are created around aggregates on borders in order. Time is written in seconds.

Joint1 of Joint Aggregation with fewer levels and parallel direct solver				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
128	76	6.9	57.1	64.1
64	40	5.2	29.0	34.2
32	18	4.5	12.7	17.3
16	15	4.0	10.1	14.2
8	15	3.8	9.9	13.8
4	15	3.5	9.6	13.1
2	18	3.2	11.3	14.5

Table 10. SA-AMG Method for Anisotropic Problems: Joint1 of joint aggregation is improved with fewer levels utilizing a parallel direct solver at the coarsest level. The number of levels is four. Time is written in seconds.

Table 11. Problem Size and Number of PEs for Isotropic Problems

Isotropic Problems		
DOF	# of PEs	Problem Domain
15625k	125	250 × 250 × 250
12500k	100	200 × 250 × 250
10000k	80	200 × 200 × 250
8000k	64	200 × 200 × 200
6000k	48	150 × 200 × 200
4500k	36	150 × 150 × 200
3375k	27	150 × 150 × 150
2250k	18	100 × 150 × 150
1500k	12	100 × 100 × 150
1000k	8	100 × 100 × 100
500k	4	50 × 100 × 100
250k	2	50 × 50 × 100

Coarsest Level Size for the 125PE case		
Aggregation Strategy	Size	# of Levels
Ind1	2457	4
Ind2	1562	4
Joint1	986	4
Joint2	1514	4
Joint3	1048	4

Table 12. Coarsest Problem Size for Isotropic Problems

When the SA-AMG method utilizes Ind1 of independent aggregation in 125PE case, one PE has 125000, 9319, 526, 20 unknowns for each level.

In Table 12, the problem size at the coarsest level is described for each aggregation strategy. The problem size is the biggest for the 125PE case and that case is written.

ICCG and SA-AMG Methods The results of the ICCG method for isotropic problems are shown in Table 13. The ICCG method's total time increases with problem size. On the other hand, Tables 14, 15, 16 17 and 18 show the result of the SA-AMG methods with various aggregations. In comparison with the ICCG method, Tables of the SA-AMG methods show that they need an almost constant time until convergence for any problem size on the grounds that problem size per PE is constant in this experiment. The SA-AMG methods work well especially for large-sized problems.

ICCG method		
# of PEs	# of Iteration	Total Time
125	602	267.9
100	590	260.0
80	537	235.2
64	481	210.5
48	463	202.4
36	417	180.8
27	364	157.8
18	340	145.7
12	296	124.9
8	241	100.9
4	207	86.6
2	165	67.3

Table 13. ICCG Method for isotropic Problems. Time is written in seconds.

Indl of Independent Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
125	23	11.1	39.1	50.2
100	23	10.9	38.4	49.4
80	22	10.8	36.8	47.8
64	22	10.8	36.6	47.5
48	22	10.7	36.6	47.4
36	22	10.7	36.2	46.9
27	21	10.4	34.4	44.8
18	21	10.2	33.5	43.8
12	21	9.8	32.5	42.4
8	21	9.4	31.6	41.1
4	21	8.8	30.7	39.6
2	21	8.4	29.6	38.0

Table 14. SA-AMG Method for Isotropic Problems: Indl of independent creates aggregates around one aggregate in order. Time is written in seconds.

Ind2 of Independent Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
125	37	10.2	61.3	71.5
100	35	10.0	56.6	66.7
80	35	10.0	56.2	66.4
64	35	10.0	55.8	65.8
48	35	10.0	55.5	65.6
36	34	9.9	53.2	63.1
27	34	9.9	52.6	62.5
18	34	9.6	51.3	61.0
12	33	9.2	48.5	57.8
8	31	8.9	44.9	53.9
4	28	8.5	39.9	48.4
2	25	8.1	34.9	43.1

Table 15. SA-AMG Method for Isotropic Problems: Ind2 of independent aggregation creates aggregates from borders. Time is written in seconds.

Joint1 of Joint Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
125	62	10.6	110.2	120.9
100	62	10.4	106.9	117.4
80	61	10.5	104.4	114.9
64	60	10.3	101.2	111.6
48	59	11.0	98.0	109.1
36	59	10.3	97.9	107.3
27	57	10.0	91.8	101.8
18	55	9.5	86.2	95.9
12	55	9.2	84.4	93.7
8	51	9.0	77.1	86.1
4	51	8.8	76.9	85.7
2	48	7.8	69.1	77.0

Table 16. SA-AMG Method for Isotropic Problems: Joint1 of joint aggregation creates aggregates in the order of the vertex number in the inner domain. Time is written in seconds.

Joint2 of Joint Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
125	28	10.9	48.4	59.4
100	28	10.7	47.8	58.6
80	28	10.6	47.2	57.9
64	27	10.5	45.1	55.7
48	27	10.5	44.7	55.2
36	27	10.2	43.9	54.2
27	26	10.5	41.6	52.2
18	25	10.9	39.1	49.3
12	25	9.7	38.2	48.0
8	23	9.4	34.7	44.1
4	22	8.9	32.0	40.9
2	19	8.4	27.0	35.5

Table 17. SA-AMG Method for Isotropic Problems: Joint2 of joint aggregation creates aggregates around one aggregate in order in the inner domain. Time is written in seconds.

Joint3 of Joint Aggregation				
# of PEs	# of Iteration	Setup Time	Solution Time	Total Time
125	55	11.6	95.8	107.5
100	55	11.4	93.6	105.1
80	55	11.3	92.9	104.2
64	54	11.2	89.4	100.7
48	54	11.2	89.1	100.3
36	53	10.9	85.2	96.2
27	50	10.7	78.6	89.4
18	51	10.6	79.2	89.9
12	49	10.2	74.4	84.7
8	44	9.8	65.9	75.8
4	50	9.8	74.5	84.3
2	37	8.9	52.4	61.4

Table 18. SA-AMG Method for Isotropic Problems: Joint3 of joint aggregation creates aggregates from borders in the inner domain. Time is written in seconds.

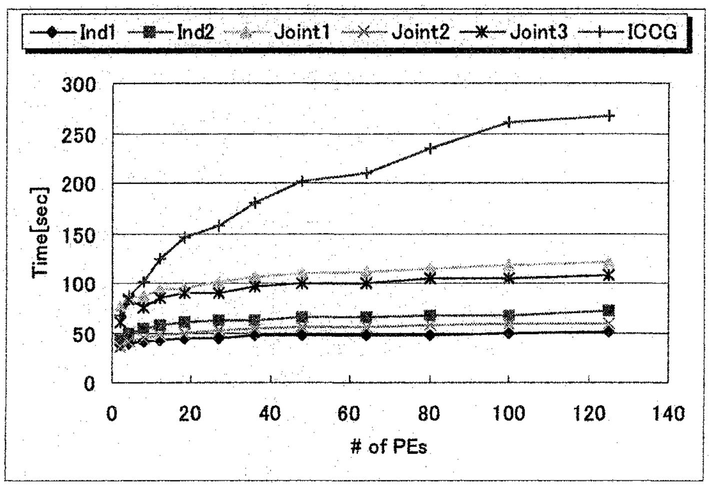


Figure 7. ICCG and SA-AMG Methods for Isotropic Problems: Ind1, Ind2, Joint1, Joint2 and Joint3 represent the aggregation methods of the SA-AMG method explained in section 3.

The SA-AMG method with Various Aggregations Unlike the anisotropic cases, Ind2 of independent aggregation from borders is inferior to Ind1 of normal independent aggregation for isotropic problems, according to Tables 14 and 15. In comparison with other aggregation strategies, Ind1 is the best strategy for these problems. The iteration numbers of Ind1 are almost constant. These numbers are between 21 and 23 for problems whose sizes are from 2.5×10^5 DOF and 1.56×10^7 DOF.

Next we consider joint aggregation strategies. Joint aggregations create aggregates on borders, then create the inner part of the domain. Joint1, Joint2 and Joint3 differ in the creation of aggregates in the inner part of the domain. Joint1 creates aggregates in the order of aggregate number, which corresponds to a greedy algorithm. Joint2 creates aggregates around one aggregate in order. Joint3 creates aggregates from borders. Tables 16, 17, 18 show the results of joint aggregation. Unlike the anisotropic problems, joint aggregations differ considerably in performance. Joint2 is the best among the three methods.

Figure 7 shows the total time for the ICCG and SA-AMG methods for each problem size. It shows the Ind1 of the normal independent aggregation strategy requires the shortest total time in our experiments. Ind1 and Joint2 of the aggregation strategies work relatively well.

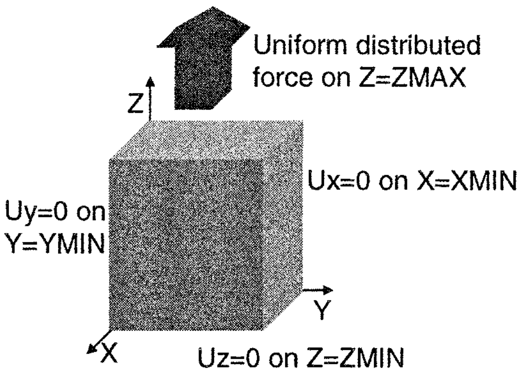


Figure 8. Cube

5.3 3-Dimensional Problems in Elasticity

The problem of the pulled cube is one of the test problems for GeoFEM. The problem is to compute the displacement U of a cube with surfaces perpendicular to the x, y and z axes. U_x represents the displacement U in the direction of the x-axis. The surface of Z=ZMAX is pulled by a uniform distributed force in the Z direction. The other boundary conditions are $U_x = 0$ on the surface of X=XMIN, $U_y = 0$ on the surface of Y=YMIN, and $U_z = 0$ on the surface of Z=ZMIN. Figure 8 shows the situation. The problem domain is decomposed into sub-domains with $35 \times 35 \times 35$ vertices, which are allocated to each PE. A vertex of finite element mesh has three DOF, corresponding to displacements in the x, y and z directions. Thus each PE deals with $35 \times 35 \times 35 \times 3$ DOF. The ICCG and SA-AMG methods are 3×3 blocked for this problem.

There is little difference in performance among parallel aggregation strategies for this problem. It seems that the problem size for each PE is too small for parallel aggregation strategies to make a difference. In this subsection, comparison between the SA-AMG and ICCG methods, and the behavior of the SA-AMG method for large problems in elasticity, are investigated. For elastic problems, all SA-AMG methods have four levels and utilize twenty iterations of the Chaotic SGS method at the coarsest level. When the SA-AMG method utilizes normal independent aggregation for the 125PE case, one PE has 42875×3, 1585×3, 80×3, and 5×3 unknowns for each level.

ICCG and SA-AMG Methods This paragraph compares the ICCG and SA-AMG methods for problems in elasticity. Figure 9 shows the

elastic problems		
DOF	# of PEs	Problem Domain
16078k	125	175 × 175 × 175
12862k	100	140 × 175 × 175
10290k	80	140 × 140 × 175
8232k	64	140 × 140 × 140
6174k	48	105 × 140 × 140
4630k	36	105 × 105 × 140
3472k	27	105 × 105 × 105
2315k	18	70 × 105 × 105
1543k	12	70 × 70 × 105
1029k	8	70 × 70 × 70
514k	4	35 × 70 × 70
257k	2	35 × 35 × 70

Table 19. Problem Size and Number of PEs for Elastic Problems

total time for the ICCG and SA-AMG methods for each problem size. ICCG method's total time increases along with the problem size. On the other hand, the SA-AMG method's total time increases little as problem size increases. This means that the number of iterations for the SA-AMG method until convergencen is almost constant. For example, the number of iterations for the SA-AMG method with Ind2 of independent aggregation is 48 for 2 PEs and 52 for 125 PEs. For this problem, the SA-AMG methods work better than ICCG method for all problem sizes. The differences in performance of aggregation strategies are small, partly because the vertex size 35 × 35 × 35 of the problem allocated to each PE is small in comparison with isotropic and anisotropic problems.

6. Summary and Conclusions

This paper compares various aggregation strategies seeking robust strategies for both isotropic and anisotropic problems. We implement the SA-AMG method which can deal with any aggregation strategy and solve anisotropic, isotropic and elastic problems. There are two contributions. First is that independent aggregation can be adapted to anisotropic problems by creating aggregates from borders and utilizing a parallel direct solver at the coarsest level. Independent aggregation is known to be a bad aggregation strategy, as is shown in [TT00]. Second is that the robust aggregation strategy is a strategy that creates aggregates around one aggregate in order after shared aggregates are created on borders. It is explained as Joint2 in section 3. The results for each type of problem are given in subsequent paragraphs.

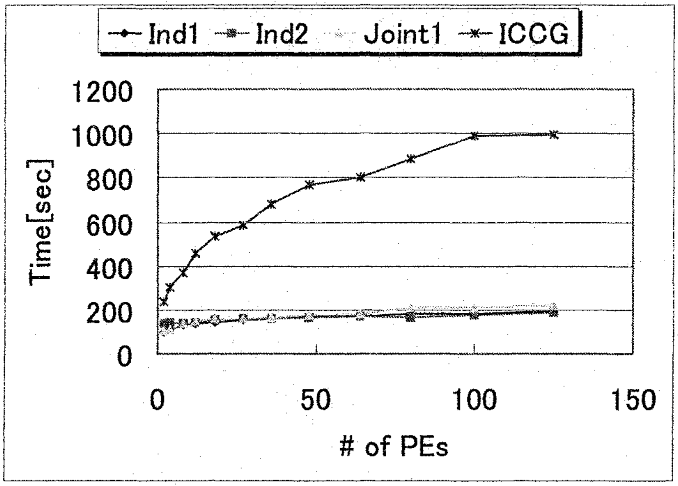


Figure 9. ICCG and SA-AMG Methods for Elastic Problems: Indl, Ind2, and Jointl represent aggregation strategies for the SA-AMG method.

For anisotropic problems, joint aggregations work more efficiently than independent aggregations, but Ind2 of independent aggregation with a parallel direct solver works as well as joint aggregations with a parallel direct solver. Methods such as Ind2 or joint aggregations create aggregates around borders first. That aggregate creation order creates no distorted aggregates around borders, which seems to be important for anisotropic problems.

For isotropic problems, Indl of independent aggregation is the best aggregation strategy in the total time. Joint aggregations differ in their performance much more than the anisotropic case. Joint2 of joint aggregation works the best of the joint aggregations. Indl and Joint2 create aggregates around one aggregate in order in the internal domain. This aggregate creation order in the internal domain seems to be important for isotropic problems. For elastic problems, there is little difference in performance. There seems to be too small a domain allocated for each PE.

References

[Ada98] M. F. Adams. A parallel maximal independent set algorithm. In *Proceedings 5th Copper mountain conference on iterative methods*, 1998.

[Geo] GeoFEM, http://www.geofem.tokyo.rist.or.jp/.

[GGJ$^+$97] Anshul Gupta, Fred Gustavson, Mahesh Joshi, George Karypis, and Vipin Kumar. Design and implementation of a scalable parallel direct solver for sparse symmetric positive definite systems. In *Proceedings of the Eighth SIAM Conference on Parallel Processing*, 3 1997.

[MPI] MPI(Message Passing Interface) Forum Web Site, http://www.mpi-forum.org/.

[Myr] Myrinet Software, http://www.myri.com/scs/.

[Nak03] K. Nakajima. *Parallel Iterative Linear Solvers with Preconditioning for Large Scale Problems.* Ph.D. dissertation, University of Tokyo, 2003.

[NNT97] K. Nakajima, H. Nakamura, and T. Tanahashi. Parallel iterative solvers with localize ILU preconditioning. In *Lecture Notes in Computer Science 1225*, pages 342–350, 1997.

[NO99] K. Nakajima and H. Okuda. Parallel iterative solvers with Localized ILU preconditioning for unstructured grids on workstation clusters. *IJCFD*, 12:315–322, 1999.

[TT00] Ray S. Tuminaro and Charles Tong. Parallel smoothed aggregation multigrid: Aggregation strategies on massively parallel machines. In *SuperComputing*, 2000.

[VBM01] Petr Vanek, Marian Brezina, and Jan Mandel. Convergence of algebraic multigrid based on smoothed aggregation. *Numerische Mathematic*, 88(3):559–579, 2001.