

Comparison of coarsening schemes for multilevel graph partitioning

Cédric Chevalier¹ and Ilya Safro²

¹ Sandia National Laboratories, Albuquerque, NM, USA, ccheval@sandia.gov*

² Argonne National Laboratory, Argonne, IL, USA, safro@mcs.anl.gov**

Abstract. Graph partitioning is a well-known optimization problem of great interest in theoretical and applied studies. Since the 1990s, many multilevel schemes have been introduced as a practical tool to solve this problem. A multilevel algorithm may be viewed as a process of graph topology learning at different scales in order to generate a better approximation for any approximation method incorporated at the uncoarsening stage in the framework. In this work we compare two multilevel frameworks based on the geometric and the algebraic multigrid schemes for the partitioning problem.

1 Introduction

Graph partitioning is a computing technique used in many fields of computer science and engineering. Applications include VLSI design, minimizing the cost of data distribution in parallel computing, optimal tasks scheduling, etc. The goal is to partition the vertices of a graph into a certain number of disjoint sets of approximately the same size, so that a cut metric is minimized. Because of the NP-hardness [17] of the problem and its practical importance, many heuristics of different nature (spectral [27], combinatorial [24,16], evolutionist [8], etc.) have been developed to provide an approximate result in a reasonable (and, one hopes, linear) computational time. However, only the introduction of the multilevel methods during the 1990s has really provided a breakthrough in efficiency and quality.

During the past two decades many attempts have been made to use multilevel strategies for solving combinatorial optimization problems [7,33]. The most frequent branches on which the multilevel algorithms have been applied are VLSI design [10,11,14], graph optimization problems [30] (with special attention to the partitioning problem [1,2,3,28,4,20,23]), and several others [9,15,32].

The main objective of a multilevel based algorithm is to create a hierarchy of problems (*coarsening*), each representing the original problem, but with fewer

* Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin company, for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-AC04-94AL85000.

** This work was supported by the Office of Advanced Scientific Computing Research, Office of Science, U.S. Department of Energy, under Contract DE-AC02-06CH11357.

degrees of freedom. For the partitioning and other graph modeled problems, this hierarchy may be viewed as a process of learning of a graph topology prior to applying any approximation method. The construction of hierarchies at different scales ends up at the level with a very small number of degrees of freedom (*coarsest level*) that allows to get a first approximation to the original problem at very large scale within an insignificant running time (even for exact algorithm) in comparison to the size of original problem. Then, the obtained approximation is sequentially projected along all levels of the hierarchy (*interpolation* or *projection*) until it reaches the original problem with some approximation for it. The projection stage can be reinforced at each level by some *refinement* algorithm that improves the quality of approximation before further projection. The projection reinforced by a refinement method is called *uncoarsening*. In terms of graph partitioning problem, the hierarchy of coarse graphs is constructed for different scales, and at each scale the approximation algorithm for this problem is applied in order to project and improve current approximations. In Figure 1, we present a small example of a multilevel framework (called *V-cycle*) for the 4-partitioning problem. For needed references and background on multilevel techniques, we refer the reader to [7].

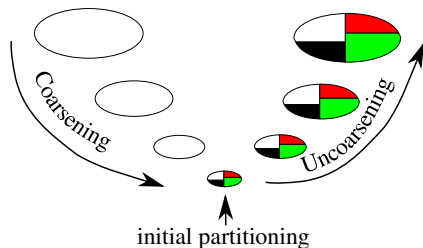


Fig. 1. Example of multilevel framework for a 4-partitioning problem. Three empty ellipses represent the three levels of the coarsening. The smallest colored by four colors ellipse corresponds to the coarsest level graph. Four colors of the graphs through the uncoarsening stage correspond to the 4-partitioning approximation.

Almost all previously developed multilevel schemes for simple graphs possess exactly the same strict coarsening. It is carried out by matching groups (usually pairs) of vertices together and representing each group with a single vertex in the coarsened space (e.g., matching [20,23], first choice [14]). Another class of multilevel schemes used for several combinatorial optimization problems is based on an *algebraic multigrid* (AMG) method [22,25,28,30]. The principal difference between these two approaches is explained in graph model terms in [30]. Because of the difficulties in performing a rigorous analysis of multilevel schemes for discrete problems, the empirical judgment of all these algorithms is usually based on the best achieved results on some test set. Multilevel algorithms consist of many algorithmic parts, and it is not easy to realize which part plays the crucial role. This paper is about the role of a coarsening scheme in a multilevel framework.

The main goal of this paper is a systematic comparison of the AMG-based scheme versus strict scheme based on heavy edge matching (HEM, adopted since 1995 and implemented in many multilevel packages) for the partitioning problem while having the uncoarsening parts (based on the popular sequential algorithm called Fiduccia-Mattheyses (FM) [16]) exactly the same in both cases. This issue still has not been studied empirically, in contrast to many other works in which a number of a more or less successful uncoarsening and postprocessing procedures have been suggested. The framework used for these experiments is SCOTCH [26], since it provides an open architecture to easily plug in different algorithms and choose between them at the runtime with the strategy string, a powerful way to dynamically choose the methods and the parameters we want to use. The AMG-based coarsening procedure was taken from [30].

2 Definitions and notation

Consider a simple weighted graph $G = (V, E)$, where $V = [1, n]$ is the set of vertices (nodes) and E is the set of edges. Denote by w_{ij} the non-negative weight of the undirected edge $ij \in E$; if $ij \notin E$, then $w_{ij} = 0$. Let v_i be a positive weight of vertex $i \in V$ and $v(A) = \sum_{i \in A} v_i$, where $A \subseteq V$.

The goal of the general graph k -partitioning problem is to find a partition of V into a family of k disjoint nonempty subsets $(\pi_p)_{1 \leq p \leq k}$, while enforcing the following:

- 1) $\sum_{i \in \pi_p \Rightarrow j \notin \pi_p} w_{ij}$ is minimized (called *interface size* or *edgcut*) and
- 2) $\max_{p \in [1, k]} \left| v(\pi_p) - \frac{v(V)}{k} \right|$ is minimized (called *balanced objective*).

It is accepted to call one subset π_p as a *part* and a family $(\pi_p)_{1 \leq p \leq k}$ as a *partition* of V . In general, two minimization objectives can often be in conflict. Thus, in most of the partitioning formulations the balance objective is restrained to be a constraint

$$\forall p \in [1, k], v(\pi_p) \leq (1 + \alpha) \cdot \frac{v(V)}{k},$$

where α is a given *imbalance factor*. In this paper, we refer to the constrained version of the problem as the graph k -partitioning problem.

A common method of solving the k -partitioning problem when $k > 2$ is to adopt a divide and conquer approach [31] that uses *recursive bisection* (or bipartitioning). To simplify the explanation, without loss of generality, we will talk about bipartitioning rather than k -partitioning.

3 Coarsening schemes

In general, any coarsening can be interpreted as a process of *aggregation* of graph nodes to define the nodes of the next coarser graph. In this paper we compare

two coarsening schemes: strict and weighted aggregations (SAG and WAG). For completeness we briefly review their description.

In SAG (also called edge contraction or matching of vertices) the nodes are blocked in small disjoint subsets, called aggregates. Two nodes i and j are usually blocked together if their coupling is *locally strong*, meaning that w_{ij} is comparable to $\min\{\max_k w_{ik}, \max_k w_{kj}\}$ (see Figure 2). In WAG, each node can be

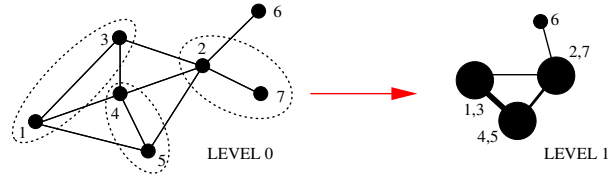


Fig. 2. Schematic demonstration of the SAG scheme. The dashed ovals correspond to the pairs of vertices at the fine level that form aggregates at the coarse level. For example, vertices "1" and "3" are aggregated into one coarse node "1,3".

divided into *fractions*. Different fractions belong to different aggregates (see Figure 3); that is, V will be covered by (presumably) small intersecting subsets of V . The nodes that belongs to more than one subset will be divided among corresponding coarse aggregates. In both cases, these aggregates will form the nodes of the *coarser level*, where they will be blocked into larger aggregates, forming the nodes of a *still coarser level*, and so on.

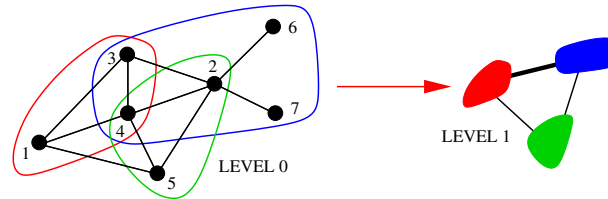


Fig. 3. Schematic demonstration of the WAG scheme. The closed curves at the left graph correspond to the subsets of vertices that form aggregates at the coarse level. These subsets are not disjoint; in other words, vertices in intersection are divided among several aggregates.

As AMG solvers have shown, *weighted*, instead of *strict*, aggregation is important in order to express the *likelihood* of nodes to belong together; these likelihoods will then accumulate at the coarser levels of the process, indicating tendencies of larger-scale aggregates to be associated to each other. SAG, in contrast, may run into a conflict between the local blocking decision and the larger-scale picture.

For both aggregation schemes, the construction of a coarse graph is divided into three stages: (a) a subset of the fine nodes is chosen to serve as the *seeds* of the aggregates (which form the nodes of the coarser level), (b) the rules for interpolation are determined, and (c) the weights of the edges between the aggregates are calculated. For simplicity, we will unify stages (a) and (b) into one stage in case of SAG. Here are the basic steps of these aggregation schemes.

SAG: coarse nodes. Visit the vertices according to some order [23] and choose an appropriate (heaviest, lightest, random, etc., see [23]) edge for making a coarse aggregate from its two endpoints i and j . The weight of a coarse aggregate will be $v_i + v_j$.

WAG: coarse nodes-(a). The construction of the set of seeds $C \subset V$ and its complement $F = V \setminus C$ is guided by the principle that each F -node should be “strongly coupled” to C . Starting from $C = \emptyset$ and $F = V$, transfer nodes from F to C until all remaining $i \in F$ satisfy

$$\sum_{j \in C} w_{ij} / \sum_{j \in V} w_{ij} \geq \Theta ,$$

where Θ is a parameter (usually $\Theta \approx 0.5$).

WAG: coarse nodes-(b). Define for each $i \in F$ a coarse neighborhood N_i consisting of C -nodes to which i is connected. Let $I(j)$ be the ordinal number in the coarse graph of the node that represents the aggregate around a seed whose ordinal number at the fine level is j . The classical AMG interpolation matrix P is defined by

$$P_{iI(j)} = \begin{cases} w_{ij} / \sum_{k \in N_i} w_{ik} & \text{for } i \in F, j \in N_i \\ 1 & \text{for } i \in C, j = i \\ 0 & \text{otherwise} \end{cases} .$$

$P_{iI(j)}$ thus represents the likelihood of i to belong to the $I(j)$ th aggregate. The volume of the p th coarse aggregate is $\sum_j v_j P_{jp}$. Note that $|N_i|$ is controlled by the parameter called *interpolation order*.

SAG: coarse edges. Introduce a weighted coarse edge between aggregates p and q created from fine pairs of vertices (i_1, i_2) and (j_1, j_2) , respectively. Then w_{pq} will accumulate all possible connections between different components of these pairs.

WAG: coarse edges. Assign the edge connecting two coarse aggregates p and q with the weight $w_{pq} = \sum_{k \neq l} P_{kp} w_{kl} P_{lq}$.

In general, both processes might be reformulated as a single algorithm. Note that, given two consecutive levels l and L , in both cases

$$\sum_{i \in G_l} v_i = \sum_{i \in G_L} v_i .$$

In contrast to the widely used SAG scheme, we are aware of two partitioning solvers [25] and [28] in which an AMG-based scheme was employed. In contrast to [25] we employed the AMG-based coarsening only once directly on the original graph. In [25] two coarsening schemes were employed: PMIS and CLJP. Some

parts of these schemes were adapted for their purposes. This solver is very successful, however, in that work was discussed a newly introduced refinement only. The process of coarsening in [28] is reinforced by compatible Gauss-Seidel relaxation [5], which improves the quality of the set of coarse-level variables, prior to deriving the coarse-level equations. The quality measure of the set of coarse-level variables is the convergence rate of F -nodes with respect to C . However, work on improving the quality of coarse-level variables and equations is in progress, and currently it is not clear whether this relaxation plays an important role for the partitioning problem.

4 Uncoarsening

In this section we will provide the details about the uncoarsening stage and several recommendations of relatively easy improvement of it.

4.1 Disaggregation

To compare the different coarsening methods described in the preceding section, we have chosen to use the same refinement techniques for all the schemes and not to develop one specifically designed for WAG.

The uncoarsening phase typically consists of two steps: the projection of the partition from the coarser graph to the finer graph and the refinement, a local optimization of the partition using the available topological information at the current level.

The projection phase is simple in the case of a strict coarsening scheme. It consists only of assigning the same part number for a fine vertex as the one assigned to its associated coarse vertex.

For a nonstrict coarsening scheme, the projection phase is more complex. We can directly project only the seeds exactly as with a strict coarsening; but for fine vertices that are not seeds, we have to do an interpolation to compute their assignments with respect to the assignments of their neighbors.

In this paper, we have focused on two simple interpolation methods. Both require computing the probability that a fine vertex belongs to a specific part. In the case of bipartitioning, only the knowledge of the probability to be in the part 0 (or 1) matters. Let us call $\mathcal{P}_0(i)$ the probability that vertex i is in part 0. With the notation introduced for coarsening, we have

$$\mathcal{P}_0(i) = \sum_{k \in N_i, I(k) \in \pi_0} P_{iI(k)} .$$

The first strategy we use is to assign a vertex i to the part 0 (resp. 1) if the probability $\mathcal{P}_0(i)$ is greater (resp. lower) than $\frac{1}{2}$.

The second strategy is to assign the part in proportion to the probability \mathcal{P}_0 .

In these two schemes, the projection and interpolation involve two consecutive loops. The first loop, the projection, browses all the seeds and set their

values to be those of their corresponding coarse vertices. The second loop, the interpolation, scans all the fine nonseed vertices and fixes them in their parts by computing \mathcal{P}_0 . The cost in time is thus $\Theta(|C| + |F| \cdot io)$, where io is the interpolation order, instead of $\Theta(|F|)$ for SAG.

The projected partition now has to be optimized by taking into account the finer knowledge of the topology. This phase is known as relaxation or refinement. In our experiments, we use one of the most popular refinement techniques, Fiduccia-Mattheyses (FM) [16]. This algorithm is popular because it is fast and allows hill-climbing optimization for the cost function. Its principle is simple: Order the vertices according to the gain in edgcut obtained if the vertex is moved to the other part; then move the vertex of highest gain, and update the gain for the neighbors and loop. It is possible that the gain can signify a decrease in the partition quality, but one hopes it can lead to a better local minimum for the edgcut. The number of degradation moves is a parameter and by default set in SCOTCH to 80.

However, we have also chosen to try a poorer refinement, for two reasons. First, to really compare the coarsening schemes, we have to avoid a too powerful refinement because it can hide some artifacts caused by the coarsening. The second reason is that a hill-climbing refinement is sometimes not available, especially in parallel algorithms. To do this poorer refinement, we continue to use FM but with a limitation during its execution: we force FM to stop if the best move will degrade the partition quality. Thus, we obtain only a gradient-like refinement.

4.2 Further improvements

As mentioned, this paper compares of two coarsening schemes given a significantly simplified, common uncoarsening stage that can be easily parallelized. However, we would like to include in this paper a list of possible further improvements of the AMG-based algorithm. These improvements were tested on the partitioning and linear ordering problems, and all have a good chance of exhibiting superior results to the basic AMG-based algorithm.

Prolongation by layers. In classical AMG schemes the initialization of fine level variables is done by a prolongation operator that is equal to the transpose of the restriction operator. In several multilevel algorithms the initialization process depends on the already-initialized variables [30,28], while the order of the initialization is determined by the strength of connection between variables and the set of already initialized variables.

Compatible relaxation. This type of strict minimization was introduced in [5] as a practical tool for improving the quality of selecting the coarse variables and consequently the relations between the fine and coarse variables. In general, this relaxation minimizes the local energy contribution of fine variables while keeping coarse variables invariant (see [30,28]).

Generating many coarse solutions. Solving the problem exactly at the coarsest level may be reinforced by producing many solutions that differ from each other and involve a lowcost partitioning simultaneously.

Cycling and linearization. One complete iteration of the algorithm is called a V -cycle, because of the order of visiting the coarse levels. Other patterns of visiting the coarse levels are also possible. A W_L -cycle was tested in the multilevel scheme for the linear ordering problems [30] and exhibited an improvement proportionally to the amount of work units it spent in comparison to the V -cycle. For both V - and W_L -cycles the *linearization* technique [29] was used to provide a current approximated solution as an initial point for further approximation.

5 Computational results

To prevent possible unexpected problems of implementation and to make a fair comparison of the two methods, we combined WAG multilevel algorithm by two separate software packages: [30] for the coarsening stage and SCOTCH for the uncoarsening. The entire strict aggregation (HEM) multilevel partitioning algorithm was taken from the SCOTCH package. The combination of two separate packages limited us in performing the bisection experiments only, since the general k -way partitioning might be produced by SCOTCH by applying a bisection method recursively. However, this limitation does not play a crucial role in understanding the general process. Usually, the quality of the k -way partitioning strongly depends on the quality of the bisectioning algorithm incorporated into the general scheme, as a small bias on the first dissection has consequence on all the next levels of bisection.

As is done in most multilevel graph partitioning implementations, the coarsening is continued until the size of the coarsest graph is more than 100 vertices. Then, an aggressive heuristic is applied to get an initial partitioning. The exact partitioning of the coarse graph does not influence the final quality if it is not used in the context of a multiprojection of different partitions.

The comparison is based on the set of real-world graphs presented in Table 1. The imbalance ratio was kept at 1% during all experiments. In order to estimate the algorithmic stability, each test was executed twenty times with different random seeds and initial reshuffling of V and E . Experiments with 100 executions per test did not provide a better estimation of a general statistical view (minimum, maximum, average, and standard deviation).

5.1 Discussion

A frequent weakness of the classical matching-based coarsening schemes may be formulated as the following observation: the results are quite unpredictable. This can be characterized by high standard deviation of the edgecuts, undesirable sensitivity to the parameters, random seed dependence, and other factors that can influence the robustness of the heuristic. In terms of the coarsening stage, this weakness can be heuristically explained by conflicts between local decisions (at the fine scale) and the global solution. In other words, by matching two vertices we assume that, according to some argument, they will share a common

<i>Graph name</i>	$ V (\times 10^3)$	$ E (\times 10^3)$	Avg. degree	Type
<i>4elt</i>	15	46	5.88	2D finite element mesh
<i>altr4</i>	26	163	12.5	Mesh, CEA-CESTA
<i>oilpan</i>	74	1762	47.78	3D stiffness matrix
<i>ship001</i>	35	2304	132	Parasol matrix
<i>tooth</i>	78	452	11.5	3D finite element mesh
<i>m14b</i>	215	1679	15.6	3D finite element mesh
<i>ocean</i>	143	410	5.71	3D finite element mesh
<i>fe_rotor</i>	100	662	13.3	3D finite element mesh
<i>598a</i>	111	742	13.37	3D finite element mesh
<i>144</i>	144	1074	15	3D finite element mesh
<i>Peku01-25</i>	13	112	17.86	Placement graph
<i>bcsstk32</i>	45	985	44.16	3D stiffness matrix
<i>thread</i>	30	2220	149.32	Parasol matrix
<i>plgr_2500_2</i>	2.5	24	19.61	Power-law graph
<i>plgr_5000_1</i>	4	6.2	3.03	Power-law graph
<i>plgr_5000_2</i>	3.8	5.2	2.72	Power-law graph
<i>plgr_5000_3</i>	4.1	6.2	3.03	Power-law graph
<i>fxm4-6</i>	19	239	25.3	Optimization problem
<i>p2p-1</i>	11	31	5.72	p2p network
<i>p2p-2</i>	11	31	5.62	p2p network

Table 1. Some of the graphs on which we ran our experiments.

property (belonging to the same π_i) and this property will be assigned to each of them at the interpolation stage as initial solution. Unfortunately, because of the NP-hardness of the partitioning problem (still) no argument can provide enough pairwise local (even with high probability) decisions for the vertices to belong to the same part. Thus, making a local decision without collecting enough global information regarding the graph can lead to the unexpectedness.

In contrast to SAG, WAG consists of two ways to prevent itself from making local decisions before collecting the global information: (a) each vertex must be connected to enough seeds and (b) the nonseed vertex might be divided between several seeds (when $io > 1$). Thus, the obtained covering of a fine graph by aggregates (like those depicted in Figure 3) is smoother and the connectivity of a coarse aggregate is better than a pair matching can ensure. However, by increasing the number of possible F-vertices divisions, the connectivity of an aggregate may be too high and can cause an increased coefficient in linear running time. Thus, it must be controlled by the *interpolation order*. Moreover, it was never observed that too high an interpolation order (more than twenty) has improved the final results significantly. It can lead to the global averaging process which result can be far from an optimality.

In Figure 4 we compare HEM and WAG with $io = 4$ (which still has a low complexity and exhibits superior results); no hill-climbing optimization was applied at the refinement in either case. Except in two peer-to-peer graphs, WAG clearly outperforms HEM, producing two times better average cuts.

This significant improvement is explained by a better conservation of a graph topology during the WAG coarsening. The main reason is that one can expect

a good AMG coarsening of the graph Laplacian when the problem is associated with, or approximated by, the problem of minimizing the quadratic functional given by $\sum_{i,j} w_{ij}(x_i - x_j)^2$, which is, in general, a natural problem that can be solved better by AMG than by geometric multigrid approaches [7]. The partitioning problem yields such an approximation while, for example, considering spectral methods [27] or quadratic programming [19].

In particular, this improvement is interesting in light of designing *parallel* graph partitioners, since many efforts are needed to obtain an efficient parallel refinement. Another important observation is that the WAG standard deviation is lower; that is, the quality of the partitions is more predictable. It can certainly reduce the number of executions of the algorithm as it works in several tools and by default in SCOTCH.

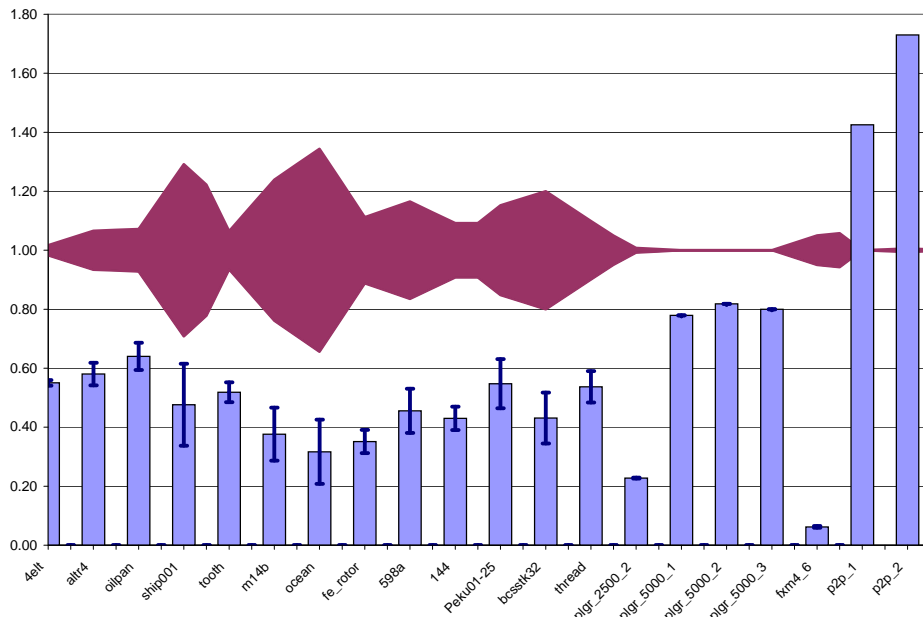


Fig. 4. Edgecut between HEM and WAG with interpolation order 4 when using a gradient refinement. The HEM average is 1. The area in purple represents the standard deviation, counted positively and negatively. The light-gray bars are the WAG average, and the dark-gray boxes the standard deviation.

The second experiment consists of applying the same WAG and HEM reinforced by FM *with hill-climbing* capabilities. The results are presented in Figure 5. For all test graphs in this case, WAG remains superior to HEM while having a lower standard deviation. More aggressive uncoarsening allows us to better exploit a graph topology and leads to the better partitions. Note that WAG clearly outperforms HEM on power-law graphs. However, in the current (not fully optimized) WAG version poorest refinement can give better results.

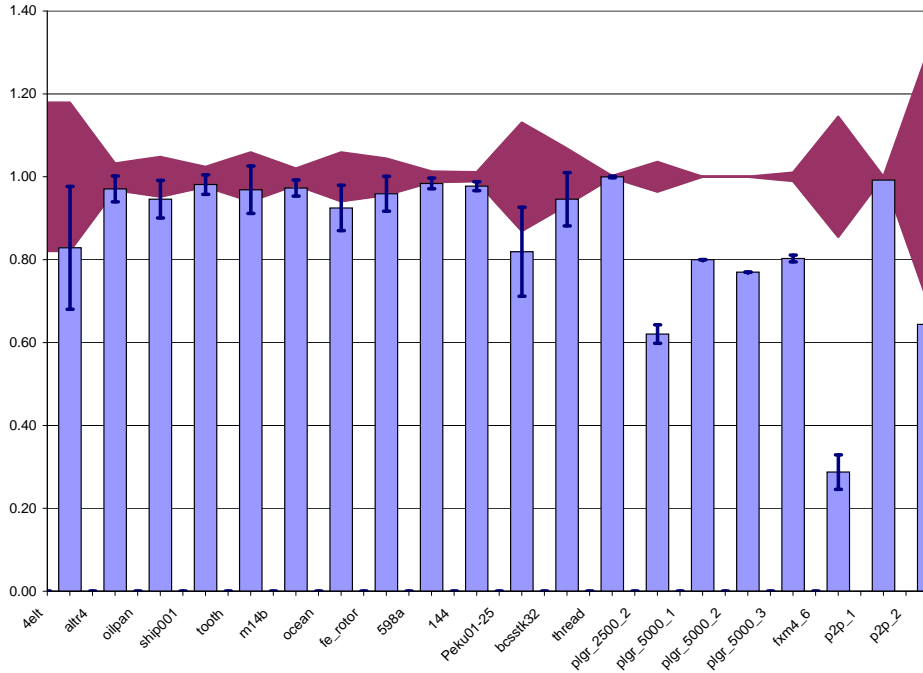


Fig. 5. Edgecut between HEM and WAG with interpolation order 4 when using a FM refinement with hill climbing. The symbols are the same as in Figure 4.

Another experiment was performed to determine the influence of the interpolation order in WAG. In the previous tests, only HEM was matching-based; that is, the size of one aggregate was limited by two. However, a study on graph partitioning of power-law graphs [1] shows that the size of the aggregate could be important. In Figure 6, WAG with interpolation order of 1 corresponds to a generic SAG, and we can observe that increasing the interpolation order usually leads to better results.

A method of *increased interpolation orders* (marked here by "inc_io") was proposed in [29]. According to this method the interpolation order must be increased as the coarse graphs become smaller. This hardly affects the total complexity of the algorithm, but it does systematically improve the obtained results since it helps to learn better a graph topology before the uncoarsening stage.

Average edgecuts for partitions computed by standard HEM and by WAG are summarized in Table 3. On our set of test graphs, WAG is on average 15% better than HEM, and worse only for the graph *p2p_1*. However, we explain this particular problem as a lack of compatible relaxation, which has to be a natural part of any AMG-based algorithm. Usually increasing the interpolation order gives better results, but it seems to be a problem in some cases when too-simplified projection and refinement are applied, since they are not designed to deal efficiently with the gain of precision of a high interpolation order.

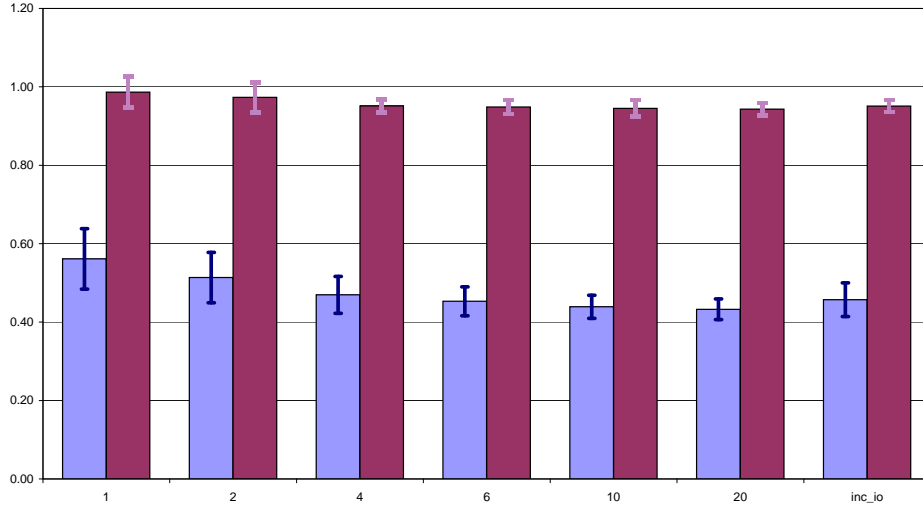


Fig. 6. Average edgecut depending on the interpolation order for WAG on 9 meshes, when using a gradient refinement without hill climbing (in blue), or a classical FM (in purple). The symbols are the same as figure 4.

Although the goal of this work was not to obtain the best known results, we present a comparison against the best known results obtained at Walshaw’s database (The graph partitioning archive. <http://staffweb.cms.gre.ac.uk/~c.walshaw/partition/>) in Table 2. Observe that the best WAG result is on average less than 0.6% from the best known partitioning. We can note that the best WAG partition is always better than the best of HEM, on average by 3% on meshes and 15% on all of our test cases. Another interesting point is that all the best results are obtained with normal FM refinement, except those for the power-law graphs. Thus, at least for this class of graphs, the improvements of uncoarsening (suggested in Section 4.2) may be interesting.

Graph name	Best known	HEM	WAG							Delta (%)	
			io=1	io=2	io=4	io=6	io=10	io=20	inc_io	best	HEM
<i>4elt</i>	138	140	138	138	138	138	138	138	138	0.00	-1.43
<i>tooth</i>	3823	4029	4100	3971	3921	3949	3947	3894	3987	1.86	-3.35
<i>m14b</i>	3826	3915	3888	3882	3860	3877	3858	3871	3863	0.84	-1.46
<i>ocean</i>	387	406	388	387	387	387	387	387	387	0.00	-4.68
<i>fe_rotor</i>	2045	2104	2085	2072	2041	2039	2053	2056	2070	-0.29	-3.09
<i>598a</i>	2388	2451	2428	2429	2414	2418	2402	2398	2415	0.42	-2.16
<i>144</i>	6479	6688	6638	6622	6596	6576	6556	6575	6600	1.19	-1.97
<i>bcsstk32</i>	4667	5009	4740	4788	4776	4757	4938	5013	4743	1.56	-5.37

Table 2. Minimum edgecut obtained on 20 runs. HEM is the standard SCOTCH matching and the *Best known* results are from Chris Walshaw’s database. All the HEM results and all the WAG results are obtained by using FM refinement with at most 80 unproductive moves for hill climbing.

Graph name	HEM	WAG							Delta (%)	
		io=1	io=2	io=4	io=6	io=10	io=20	inc_io	HEM	
<i>4elt</i>	170	152	154	141	142	140	142	142	-14.70	
<i>altr4</i>	1656	1638	1619	1607	1593	1587	1593	1619	-2.88	
<i>oilpan</i>	9433	9533	9153	8923	8974	8744	8771	8854	-4.66	
<i>ship001</i>	17052	16787	16754	16733	16720	16616	16562	16770	-2.03	
<i>tooth</i>	4346	4418	4289	4211	4152	4129	4114	4133	-3.21	
<i>m14b</i>	4029	4043	3968	3920	3920	3913	3907	3914	-2.20	
<i>ocean</i>	427	420	417	395	393	392	391	395	-6.33	
<i>fe_rotor</i>	2205	2147	2129	2115	2112	2160	2107	2123	-3.51	
<i>598a</i>	2484	2478	2458	2444	2438	2424	2416	2440	-1.67	
<i>144</i>	6826	6909	6763	6671	6637	6632	6621	6672	-1.83	
<i>Peku01-25</i>	8305	7008	7049	6802	6798	6742	6711	6819	-17.55	
<i>bcsstk32</i>	5588	5576	5425	5286	5067	5154	5176	5066	-6.06	
<i>thread</i>	55933	55966	55899	55917	55970	55898	55943	55990	0.01	
<i>plgr_2500_2</i>	6908	2308	1605	1920	2117	2588	3294	2018	-67.22	
<i>plgr_5000_1</i>	946	771	775	757	751	752	740	762	-19.84	
<i>plgr_5000_2</i>	627	491	496	483	478	466	466	482	-23.41	
<i>plgr_5000_3</i>	939	772	774	754	751	747	747	753	-19.37	
<i>fxm4_6</i>	1639	511	478	471	471	484	518	471	-70.33	
<i>p2p_1</i>	1951	1944	1985	1936	2326	2447	2221	2254	10.65	
<i>p2p_2</i>	3762	1981	2123	2423	2451	2261	2663	2240	-38.71	

Table 3. Average edgecut over 20 run with hill-climbing FM for HEM and WAG with various interpolation orders. Delta is the difference between HEM and the average of WAG, in percent. The numbers in bold correspond to the best average edgecut for a graph.

6 Conclusions

This paper compares two coarsening schemes in the context of graph partitioning. As a main result of this work, we recommend the adoption of WAG instead of classical HEM because of its higher ability of graph topology learning prior to the uncoarsening stage. In general, WAG improves the quality of the partitions and thus provides a better chance of finding a good approximation.

Another interesting result is that WAG schemes seem more robust than SAG coarsenings since they provide good-quality results even with a poor refinement. Since parallel implementations in algebraic multigrid solvers [21,18] are very scalable and since the refinement is often poor in parallel, WAG appears to be an ideal candidate to design highly scalable efficient parallel graph partitioning tools.

The framework we use allowed us to combine different coarsenings with different uncoarsenings. For example, we have done several experiments with a band-FM refinement [12], which, despite the simplicity of our projection, worked well with similar results. WAG allows one to obtain superior results with several different tested methods that are of great interest for parallel implementation [13].

The partitionings obtained during our experiments certainly may be improved by using more sophisticated projection and relaxation methods at the refinement, as mentioned in 4.2.

Acknowledgement.

This work was funded by the CSCAPES institute, a DOE project. We express our gratitude to Dr. Erik Boman for useful advice.

References

1. Amine Abou-Rjeili and George Karypis. Multilevel algorithms for partitioning power-law graphs. In *IPDPS*, 2006.
2. Charles J. Alpert, Jen-Hsin Huang, and Andrew B. Kahng. Multilevel circuit partitioning. In *Design Automation Conference*, pages 530–533, 1997.
3. R. Banos, C. Gil, J. Ortega, and F.G. Montoya. Multilevel heuristic algorithm for graph partitioning. In *Applications of Evolutionary Computing*, pages 143–153, 2003.
4. Stephen T. Barnard and Horst D. Simon. A fast multilevel implementation of recursive spectral bisection for partitioning unstructured problems. *Concurrency: Practice and Experience*, 6:101–107, 1994.
5. A. Brandt. General highly accurate algebraic coarsening. *Electronic Trans. Num. Anal.* 10 (2000) 1-20, 2000.
6. A. Brandt, S. McCormick, and J. Ruge. Algebraic multigrid (AMG) for automatic multigrid solution with application to geodetic computations. Technical report, Institute for Computational Studies, Fort Collins, CO, POB 1852, 1982.
7. A. Brandt and D. Ron. Chapter 1 : Multigrid solvers and multilevel optimization strategies. In J. Cong and J. R. Shinnerl, editors, *Multilevel Optimization and VLSICAD*. Kluwer, 2003.
8. T. N. Bui and B. R. Moon. Genetic algorithm and graph partitioning. *IEEE Trans. Comput.*, 45(7):841–855, 1996.
9. U. Catalyurek and C. Aykanat. Decomposing irregularly sparse matrices for parallel matrix-vector multiplications. *Lecture Notes in Computer Science*, 1117:75–86, 1996.
10. Tony F. Chan, Jason Cong, Michail Romesis, Joseph R. Shinnerl, Kenton Sze, and Min Xie. mpl6: a robust multilevel mixed-size placement engine. In *ISPD*, pages 227–229, 2005.
11. C. Chang, J. Cong, D. Pan, and X. Yuan. Multilevel global placement with congestion control. *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Systems*, 22:395–409, 2003.
12. C. Chevalier and F. Pellegrini. Improvement of the efficiency of genetic algorithms for scalable parallel graph partitioning in a multi-level framework. In *Proc. Eurompar, Dresden, LNCS 4128*, pages 243–252, September 2006.
13. C. Chevalier and F. Pellegrini. Pt-scotch: A tool for efficient parallel graph ordering. *Parallel Comput.*, 34(6-8):318–331, 2008.
14. J. Cong and J. R. Shinnerl, editors. *Multilevel Optimization and VLSICAD*. Kluwer, 2003.
15. Karen Devine, Erik Boman, Robert Heaphy, Bruce Hendrickson, and Courtenay Vaughan. Zoltan data management services for parallel dynamic applications. *Computing in Science and Engineering*, 4(2):90–97, 2002.
16. C. M. Fiduccia and R. M. Mattheyses. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Design Automation Conference*, pages 175–181. IEEE, 1982.

17. M. R. Garey, D. S. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problems. *Theoretical Computer Science*, 1:237–267, 1976.
18. M.W. Gee, C.M. Siefert, J.J. Hu, R.S. Tuminaro, and M.G. Sala. ML 5.0 smoothed aggregation user’s guide. Technical Report SAND2006-2649, Sandia National Laboratories, 2006.
19. William W. Hager and Yaroslav Krylyuk. Graph partitioning and continuous quadratic programming. *SIAM J. Discret. Math.*, 12(4):500–523, 1999.
20. Bruce Hendrickson and Robert W. Leland. A multi-level algorithm for partitioning graphs. In *Supercomputing*, 1995.
21. Van Emden Henson and Ulrike Meier Yang. Boomerang: a parallel algebraic multigrid solver and preconditioner. *Appl. Numer. Math.*, 41(1):155–177, 2002.
22. Y. F. Hu and J. A. Scott. A multilevel algorithm for wavefront reduction. *SIAM J. Scientific Computing*, 23:2000–031, 2001.
23. G. Karypis and V. Kumar. A fast and high quality multilevel scheme for partitioning irregular graphs. Technical Report 95-035, University of Minnesota, June 1995.
24. B. W. Kernighan and S. Lin. An efficient heuristic procedure for partitioning graphs. *BELL System Technical Journal*, pages 291–307, February 1970.
25. Henning Meyerhenke, Burkhard Monien, and Thomas Sauerwald. A new diffusion-based multilevel algorithm for computing graph partitions of very high quality. In *Proc. 22nd International Parallel and Distributed Processing Symposium, (IPDPS’08)*. IEEE Computer Society, 2008. Best Algorithms Paper Award.
26. SCOTCH: Static mapping, graph partitioning, and sparse matrix block ordering package. <http://www.labri.fr/~pelegrin/scotch/>.
27. A. Pothen, H. D. Simon, and K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal of Matrix Analysis*, 11(3):430–452, July 1990.
28. D. Ron, S. Wishko-Stern, and A. Brandt. An algebraic multigrid based algorithm for bisectioning general graphs. Technical Report MCS05-01, Department of Computer Science and Applied Mathematics, The Weizmann Institute of Science, 2005.
29. I. Safro, D. Ron, and A. Brandt. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms*, 60(1):24–41, 2006.
30. I. Safro, D. Ron, and A. Brandt. Multilevel algorithms for linear ordering problems. *Journal of experimental algorithmics (in press)*, 2007.
31. Horst D. Simon and Shang hua Teng. How good is recursive bisection. *SIAM J. Sci. Comput.*, 18:1436–1445, 1997.
32. C. Walshaw. A multilevel approach to the travelling salesman problem. Tech. Rep. 00/IM/63, Comp. Math. Sci., Univ. Greenwich, London SE10 9LS, UK, August 2000.
33. C. Walshaw. Multilevel refinement for combinatorial optimisation problems. *Annals Oper. Res.*, 131:325–372, 2004.