Advanced Coarsening Schemes for Graph Partitioning

ILYA SAFRO, Clemson University
PETER SANDERS and CHRISTIAN SCHULZ, Karlsruhe Institute of Technology

The graph partitioning problem is widely used and studied in many practical and theoretical applications. Today, multilevel strategies represent one of the most effective and efficient generic frameworks for solving this problem on large-scale graphs. Most of the attention in designing multilevel partitioning frameworks has been on the refinement phase. In this work, we focus on the coarsening phase, which is responsible for creating structures similar to the original but smaller graphs. We compare different matching- and AMG-based coarsening schemes, experiment with the algebraic distance between nodes, and demonstrate computational results on several classes of graphs that emphasize the running time and quality advantages of different coarsening schemes.

Categories and Subject Descriptors: F.2.1 [**Numerical Algorithms and Problems**]; G.1.6 [**Optimization**]; G.2.2 [**Graph Theory**]

General Terms: Algorithms, Graph Partitioning, Algebraic Multigrid

Additional Key Words and Phrases: Coarsening, uncoarsening, refinement, multilevel algorithm, computational optimization, algebraic distance

ACM Reference Format:
Ilya Safro, Peter Sanders, and Christian Schulz. 2014. Advanced coarsening schemes for graph partitioning. ACM J. Exp. Algor. 19, 2, Article 2.2 (December 2014), 24 pages.
DOI: http://dx.doi.org/10.1145/2670338

1. INTRODUCTION

Graph partitioning is a class of problems used in many fields of computer science and engineering. Applications include VLSI design, load balancing for parallel computations, network analysis, and optimal scheduling. 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. This problem is NP-complete even for several restricted classes of graphs, and there is no constant factor approximation algorithm for general graphs [Bui and Jones 1992]. In this article, we focus on a version of the problem that constrains the maximum block size to $(1 + \epsilon)$ times the average block size and tries to minimize the total cut size, namely, the number of edges that run between blocks.

Notable developments in exact algorithms for graph partitioning have been done in the areas of linear and quadratic programming [Hager et al. 2013; Hager and Krylyuk 1999; Karisch et al. 2000; Fan and Pardalos 2010]. Because of the practical importance, many heuristics of a different nature (spectral [Pothen et al. 1990], combinatorial

This work is partially supported by DFG SA 933/10-1 and CSCAPES Institute, a DOE project.
Authors' addresses: I. Safro, School of Computing, McAdams Hall, Clemson University, Clemson SC 29634, USA; email: isafro@clemson.edu; P. Sanders and C. Shultz, Department of Informatics, Karlsruhe Institute of Technology, Am Fasanengarten 5, Karlsruhe 76131, Germany.

DOI: http://dx.doi.org/10.1145/2670338

[Fiduccia and Mattheyses 1982], evolutionist [Bui and Moon 1996; Sanders and Schulz 2012], etc.) have been developed to provide an approximate result in a reasonable (and, one hopes, linear) computational time. We refer the reader to Fjallstrom [1998], Schloegel et al. [2000], and Walshaw and Cross [2007] for more material. However, only the introduction of the general-purpose multilevel methods during the 1990s has provided a breakthrough in efficiency and quality. The basic idea can be traced back to multigrid solvers for solving elliptic partial differential equations [Trottenberg and Schuller 2001], but more recent practical methods particularly are based on mostly graph-theoretic aspects of edge contraction and local search. Well-known software packages based on this approach include Jostle [Walshaw and Cross 2007], Metis [Schloegel et al. 2000], DiBaP [Meyerhenke et al. 2008], and Scotch [Pellegrini n.d.].

A multilevel algorithm consists of two main phases: *coarsening*, where the problem instance is gradually mapped to smaller ones to reduce the original complexity (i.e., the graph underlying the problem is compressed), and *uncoarsening*, where the solution for the original instance is constructed by using the information inherited from the solutions created at the next coarser levels. So far, most of the attention in designing the multilevel partitioning frameworks has been on the uncoarsening phase. In this work, we focus on the coarsening phase, which is responsible for creating graphs that are smaller than but structurally similar to the given graph. We compare different coarsening schemes, introduce new elements to them, and demonstrate computational results. For this purpose, different coarsening schemes are integrated into the Karlsruhe Fast Flow Partitioner (KaFFPa), a graph partitioning framework [Sanders and Schulz 2011].

The article is organized as follows. We begin in Section 2 by introducing notation and the multilevel approach. In Section 3, we describe different coarsening schemes, including a novel algebraic, multigrid-inspired balanced coarsening scheme and matching-based coarsening schemes, as well as new measures for connectivity. We present a large experimental evaluation in Section 4 on graphs arising in real-world applications and on graphs that are specifically designed to be hard for multilevel algorithms.

2. PRELIMINARIES

Consider an undirected graph $G = (V, E, c, \omega)$ with edge weights[1] $\omega : E \to \mathbb{R}_{>0}$, node weights $c : V \to \mathbb{R}_{\geq 0}$, $n = |V|$, and $m = |E|$. We extend c and ω to sets; in other words,

$$c(V') := \sum_{v \in V'} c(v) \text{ and } \omega(E') := \sum_{e \in E'} \omega(e).$$

Here, $\Gamma(v) := \{u : \{v, u\} \in E\}$ denotes the neighbors of v. We are looking for *blocks* of nodes $V_1, \ldots, V_k$ that partition V, namely, $V_1 \cup \cdots \cup V_k = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$. A *balance constraint* demands that

$$\forall i \in \{1..k\} : c(V_i) \leq L_{\max} := (1 + \epsilon)c(V)/k + \max_{v \in V} c(v)$$

for some parameter ϵ. The last term in this equation arises because each node is atomic, and therefore a deviation of the heaviest node has to be allowed. Note that in the unweighted case, this constraint becomes $\forall i \in \{1..k\} : |V_i| \leq L_{\max} := (1 + \epsilon)\lceil |V|/k \rceil$, which is widely used by partitioning packages and in the constraint used in the Walshaw's benchmark [Soper et al. 2004]. The objective is to minimize the total *cut*

$$\sum_{i<j} \omega(E_{ij}),$$

[1]Subscripts will be used for a short notation; in other words, ω_{ij} corresponds to the weight of $\{i, j\} \in E$.

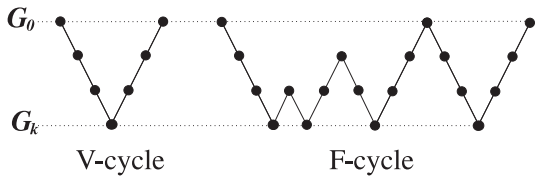


Fig. 1. V- and F-cycles schemes.

where

$$E_{ij} := \{\{u, v\} \in E : u \in V_i, v \in V_j\}.$$

A vertex $v \in V_i$ that has a neighbor $w \in V_j, i \neq j$ is a boundary vertex. We denote by $\mathsf{nnzr}(A, i)$ and $\mathsf{nnzc}(A, i)$ the number of nonzero entries in the ith row or column of a matrix A, respectively.

A matching $M \subseteq E$ is a set of edges that do not share any common nodes; that is, the graph (V, M) has a maximum degree of one. *Contracting* an edge $\{u, v\}$ means replacing the nodes u and v by a new node x connected to the former neighbors of u and v. We set $c(x) = c(u) + c(v)$ so that the weight of a node at each level is the total weight of the nodes it is representing in the original graph. If replacing edges of the form $\{u, w\}$, $\{v, w\}$ would generate two parallel edges $\{x, w\}$, we insert a single edge with $\omega(\{x, w\}) = \omega(\{u, w\}) + \omega(\{v, w\})$. *Uncontracting* an edge e undoes its contraction. A dominating set for a graph is a subset of nodes $S \subset V$ such that every vertex in $V \setminus S$ is adjacent to some node in S.

2.1. Multilevel Graph Partitioning

In the multilevel framework, we construct a hierarchy of decreasing-size graphs, $G_0, G_1, \ldots, G_k$, by *coarsening*, starting from the given graph G_0 such that each next-coarser graph G_i reflects basic properties of the previous graph G_{i-1}. At the coarsest level, G_k is partitioned by a hybrid of external solvers, and starting from the $(k-1)$th level the solution is projected gradually (level by level) to the finest level. Each projection is followed by the *refinement*, which moves nodes between the blocks to reduce the size of the cut. This entire process is called a *V-cycle* (see Figure 1). KaFFPa [Sanders and Schulz 2011] extended the concept of *iterated multilevel algorithms*, which was introduced for graph partitioning by Walshaw [2004]. The main idea is to iterate the multilevel-scheme using different random seeds for coarsening and uncoarsening. This ensures nondecreased partition quality since the refinement algorithms of KaFFPa guarantee no worsening. In this article, for the purpose of comparison, we consider also F-cycles [Sanders and Schulz 2011] (see Figure 1) as a potentially stronger and slower version of the multilevel framework for the graph partitioning problem. The detailed description of F-cycles for the multilevel graph partitioning framework can be found in Sanders and Schulz [2011].

3. COARSENING SCHEMES

One of the most important concerns of multilevel schemes is a measure of the connection strength between vertices. For matching-based coarsening schemes, experiments indicate that more sophisticated *edge rating* functions are superior to edge weight as a criterion for the matching algorithm [Holtgrewe et al. 2010]. To be more precise, first the edges get rated using a rating function that indicates how much sense it makes to contract an edge. Then a matching algorithm tries to maximize the sum of the ratings

of the edges to be contracted. The default configurations of KaFFPa employ the ratings

$$\text{expansion}^{*2}(\{u,v\}) := \omega(\{u,v\})^2/c(u)c(v), \text{ and}$$
$$\text{innerOuter}(\{u,v\}) := \omega(\{u,v\})/(\text{Out}(v)+\text{Out}(u)-2\omega(u,v)),$$

where $\text{Out}(v) := \sum_{x\in\Gamma(v)} \omega(\{v,x\})$, since they yielded the best results in Holtgrewe et al. [2010].

3.1. Algebraic Distance for Graph Partitioning

The notion of algebraic distance introduced in Ron et al. [2011] and Chen and Safro [2011] is based on the principle of obtaining low-residual error components used in the Bootstrap AMG [Brandt 2001]. When a priori knowledge of the nature of this error is not available, slightly relaxed random vectors are used to approximate it. This principle was used for linear ordering problems to distinguish between local and global edges [Ron et al. 2011]. The main difference between the k-partitioning problem and other (not necessarily combinatorial) problems for which the algebraic distance has been tested so far is the balance constraint. For many instances, it is important to keep the coarsening balanced; otherwise, even though the structural information will be captured by a sophisticated coarsening procedure, most of the actual computational work that constructs the approximate solution will be done by the refinement iterations. Bounding the number of refinement iterations may dramatically decrease its quality. Thus, a volume-normalized algebraic distance is introduced to take into account the balancing of vertices.

Given the Laplacian of a graph $L = D - W$, where W is a weighted adjacency matrix of a graph and D is the diagonal matrix with entries $D_{ii} = \sum_j \omega_{ij}$, we define its volume-normalized version denoted by $\tilde{L} = \tilde{D} - \tilde{W}$ based on volume-normalized edge weights $\tilde{\omega}_{ij} = \omega_{ij}/\sqrt{c(i)c(j)}$. We define an iteration matrix H for Jacobi overrelaxation (for $\alpha = 1/2$, also known as a lazy random-walk matrix) as

$$H = (1-\alpha)I + \alpha\tilde{D}^{-1}\tilde{W},$$

where $0 \le \alpha \le 1$. The algebraic distance coupling ρ_{ij} is defined as

$$\rho_{ij} = \left(\sum_{r=1}^{R} |\chi_i^{(k,r)} - \chi_j^{(k,r)}|^2\right)^{\frac{1}{2}},$$

where

$$\chi^{(k,r)} = H^k \chi^{(0,r)} \tag{1}$$

is a relaxed randomly initialized test vector (i.e., $\chi^{(0,r)}$ is a random vector sampled over [–1/2, 1/2]), R is the number of test vectors, and k is the number of iterations. In our experimental settings, we set $\alpha = 0.5$, $R = 5$, and $k = 20$. Choosing slightly different parameters is not critical for experimental results. It is important, however, to mention that setting $\alpha = 1$ leads to Jacobi iterations that are not convergent for bipartite graphs. In contrast to the parallelizable Jacobi overrelaxation iterator, one can also use a faster Gauss-Seidel iterator losing at the same time an easy parallelization of matrix-vector multiplication iterations of (1). We never observed that adding more test vectors ($R > 5$) can either improve or worsen the results. At the same time, decreasing R can make an initial random choice of test vectors too influential if k is small. These parameters are described in detail in Ron et al. [2011] and Chen and Safro [2011], where additional evidence of their robustness is presented for the linear arrangement problems.

3.2. Coarsening

To the best of our knowledge, the existing multilevel algorithms for combinatorial optimization problems (such as k-partitioning, linear ordering, clustering, and segmentation) can be divided into two classes: contraction-based schemes [Sanders and Schulz 2011; Dhillon 2005; Karypis and Kumar 1995] (including contractions of small subsets of nodes [Bartel et al. 2010]) and algebraic multigrid (AMG)-inspired schemes [Hu and Scott 2001; Sharon et al. 2000; Ron et al. 2005; Safro et al. 2006]. For the completeness of this work, we refer the reader to Appendix A, where we present a brief background on AMG.

3.2.1. AMG-Inspired Coarsening. One of the most traditional approaches for derivation of the coarse systems in AMG is the Galerkin operator [Trottenberg and Schuller 2001], which projects the fine system of equations to the coarser scale. In the context of graphs, this projection is defined as

$$L_c = P^T L_f P, \tag{2}$$

where L_f and L_c are the Laplacians of fine and coarse graphs $G_f = (V_f, E_f)$ and $G_c = (V_c, E_c)$, respectively. The (i, J)th entry of projection matrix $P \in \mathbb{R}^{|V_f| \times |V_c|}$ represents the strength of the connection between fine node i and coarse node J. The entries of P, referred to as interpolation weights, describe both the coarse-to-fine and fine-to-coarse relations between nodes.

The coarsening begins by selecting a dominating set of (seed or coarse) nodes $C \subset V_f$ such that all other (fine) nodes in $F = V_f \setminus C$ are strongly coupled to C. This selection can be done by traversing all nodes and leaving node i in F (initially $F = V_f$, and $C = \emptyset$) that satisfies

$$\sum_{j \in C} 1/\rho_{ij} \geq \Theta \cdot \sum_{j \in V_f} 1/\rho_{ij}, \tag{3}$$

where Θ is a parameter of coupling strength. As in AMG-based approaches for linear ordering problems [Safro et al. 2008], we observed that the order in which V_f is traversed does play an important role in reducing the dependence on random seeds. The nodes are traversed in future volume order, which measures how large an aggregate seeded by $i \in F$ might grow. Namely, we define node future volume ν_i

$$\nu_i = c_i + \sum_{ij \in E} c_j \frac{\rho_{ij}^{-1}}{\sum_{jk \in E} \rho_{jk}^{-1}}$$

(for details, see Ron et al. [2011]).

The Galerkin operator construction differs from other AMG-based approaches for combinatorial optimization problems. Balancing constraints of the partitioning problem requires a limited number of fine-to-coarse attractions between $i \in C$ (ith column in P) and its neighbors from F (nonzero entries in the ith column in P). In particular, this is important for graphs where the number of high-degree nodes in C is smaller than the number of parts in the desired partition. Another well-known problem of AMG that can affect the performance of the solver is the complexity of coarse levels. Consideration of the algebraic distance makes it possible to minimize the order of interpolation (the number of fractions to which a node from F can be divided) to 1 or 2 only [Ron et al. 2011]. Algorithm 1 summarizes and Figure 2 illustrates the construction of P.

Algorithm 1 can be viewed as simplified version of bootstrap AMG [Brandt 2001] with the additional restriction on future volume of aggregates and adaptive interpolation order. $P_{iI(j)}$ thus represents the likelihood of i belonging to the $I(j)$th aggregate. The edge connecting two coarse aggregates p and q is assigned with the weight

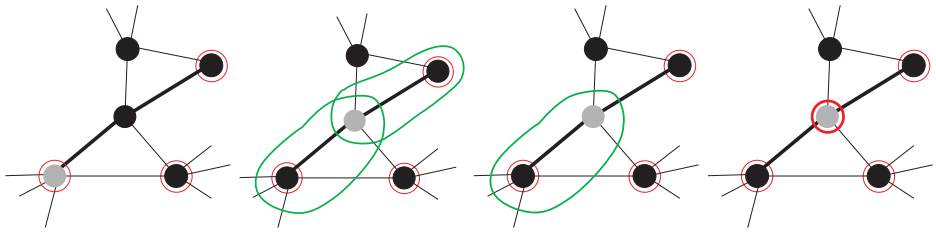


Fig. 2. Different cases for the construction of interpolation weights P. Seed vertices are surrounded by a circle. (a) For seed vertices i, we set $P_{iI(j)}$. (b) The algorithm first tries to split a vertex between two seed vertices such that the seed vertices will not be overloaded when the grey vertex is split among them. (c) If this is not successful, the algorithm tries to find a seed vertex that is not overloaded if the grey vertex is aggregated with it. (d) If (b) and (c) are not successful, then the grey vertex is added to the set of seed vertices.

ALGORITHM 1: Interpolation weights for P

input: $G, i \in V_f, P$

1 **if** $i \in C$ **then**
2 $\quad P_{iI(i)} \leftarrow 1$;
3 **else**
4 $\quad l \leftarrow$ list of at most κ algebraically strongest connections of i to C;
5 $\quad \{e_1, e_2\} \leftarrow$ algebraically strongest pair of edges (according to $\rho_{e_1} + \rho_{e_2}$) in l such that the corresponding C-neighbors are not overloaded if i is divided between them;
6 $\quad$ **if** $\{e_1, e_2\} \neq \emptyset$ **then**
7 $\quad\quad l \leftarrow \{e_1, e_2\}$
8 $\quad$ **else**
9 $\quad\quad e_1 \leftarrow$ algebraically strongest connection of i to C such that the corresponding C-neighbor is not overloaded if i is aggregated with it;
10 $\quad\quad l \leftarrow \{e_1\}$;
11 $\quad$ **if** l *is empty* **then**
12 $\quad\quad$ move i to C
13 $\quad$ **else**
14 $\quad\quad N_i^c \leftarrow C$-neighbors of i that are adjacent to edges in l;
15 $\quad\quad P_{iI(j)} \leftarrow 1/(\rho_{ij} \cdot \sum_{k \in N_i^c} 1/\rho_{ik})$ for all $j \in N_i^c$;
16 $\quad\quad$ update future volumes of all $j \in N_i^c$;

$w_{pq} = \sum_{k \neq l} P_{kp} w_{kl} P_{lq}$. The volume of the pth coarse aggregate is $\sum_j c(j) P_{jp}$. We emphasize the property of adaptivity of C (line 12 in Algorithm 1), which is updated if the balancing of aggregates is not satisfied.

We mention the difference between our AMG scheme and the weighted aggregation (WAG) scheme in Chevalier and Safro [2009]. The common principle that works in both schemes is based on the division of F-nodes between their C-neighbors. However, two critical components are missing in Chevalier and Safro: (1) the algebraic distance that forms both the set of seeds and the interpolation operator, and (2) the weight-balancing algorithmic component when aggregates are created, namely, operator P in Chevalier and Safro is created as in classical AMG schemes. One important disadvantage of Chevalier and Safro is a relatively high density of coarse levels, which is eliminated with introduction of the algebraic distance. This was achieved by reducing the order of interpolation to 1 or 2. The balancing factor played an important role in reducing

the running time of the algorithm. Recently introduced max-flow/min-cut refinement leads to noticeably better results than FM/KL heuristics (explained in Section 3.4). In contrast to simple FM/KL swaps, however, its complexity becomes higher if the aggregates are unbalanced with respect to the maximum size of one part. Applying this refinement with unbalanced WAG can significantly increase the total running time of the solver or lead to weak solutions if the refinement is terminated before it finds a good local minimum. Overall, the performance of our AMG scheme differs significantly from what we observed with WAG.

3.2.2. Matching-Based Coarsening. Another coarsening framework, which is more popular because of its simplicity and faster performance, is the *matching-based scheme*. In this scheme, a coarse graph is constructed by using contractions derived from a preprocessed edge matching. This scheme represents a special case of PL_fP^T, in which $\mathsf{nnzr}(P, r) = 1$ for all rows r in P and $1 \leq \mathsf{nnzc}(P, c) \leq 2$ for all columns c in P.

Global paths algorithm. The Global Paths algorithm (GPA) was proposed in Maue and Sanders [2007] as a synthesis of Greedy and Path Growing algorithms [Drake and Hougardy 2003]. Similar to the Greedy approach, GPA scans the edges in order of decreasing weight (or rating), but rather than immediately building a matching, it first constructs a collection of paths and even-length cycles. To be more precise, these paths initially contain no edges. While scanning the edges, the set is then extended by successively adding applicable edges. An edge is called *applicable* if it connects two endpoints of different paths or the two endpoints of an odd-length path. Afterward, optimal solutions are computed for each of these paths and cycles using dynamic programming. KaFFPaStrong [Sanders and Schulz 2011] employs innerOuter on the first level of the hierarchy since expansion^{*2} evaluates to one on unweighted graphs. Afterward, it uses expansion^{*2}.

RandomGPA algorithm. This algorithm is used by the classic KaFFPaEco configuration. It is a synthesis of the most simple random matching algorithm and the GPA algorithm. To be more precise, this matching algorithm depends on the number of blocks in which the graph has to be partitioned. It matches the first $\max\{2, 7 - \log k\}$ levels using the random matching algorithm and switches to the GPA algorithm afterward. The random matching algorithm traverses the nodes in a random order, and if the current node is not already matched, it chooses a random unmatched neighbor for the matching. KaFFPaEco employs expansion^{*2} as a rating function as soon as it uses GPA.

3.3. The Coarsest Level

Contraction is stopped when the graph is small enough to be partitioned by some other expensive algorithm. We use the same initial partitioning scheme as in KaFFPa [Sanders and Schulz 2011], namely, the libraries Scotch and Metis for initial partitioning. For AMG, some modifications have to be made, as Scotch and Metis cannot deal with fractional numbers and Metis expects $\omega_{ij} \geq 1$. To overcome this implementational problem, we perform the following two steps. First, we divide each edge weight of the coarsest graph by the smallest edge weight that occurred on that level. This step assures edge weights larger than or equal to one without skewing the graph partitioning problem for the library used. Second, we get rid of the fractional edge weights using randomized rounding. Let $e \in E$ be an edge with fractional edge weight. We then obtain an integer edge weight $\tilde{\omega}(e)$ by flipping a coin with probabilities $\mathcal{P}(\text{head}) = \omega(e) - \lfloor \omega(e) \rfloor$, $\mathcal{P}(\text{tail}) = 1 - \mathcal{P}(\text{head})$. In the case of heads, we set the edge weight $\tilde{\omega}(e)$ to $\lceil \omega(e) \rceil$; otherwise, we set it to $\lfloor \omega(e) \rfloor$. This way we can assure that the value of the cut in the graph $\tilde{G} = (V_k, E_k, \tilde{\omega})$ produced by the external initial partitioning algorithm is close to the real cut value in G.

3.4. Uncoarsening

Recall that uncoarsening undoes contraction. For AMG-based coarsening, this means that fine nodes have to be assigned to blocks of the partition of the finer graph in the hierarchy. We assign a fine node v to the block that minimizes $\text{cut}_B \cdot p_B(v)$, where cut_B is the cut after v would be assigned to block B and $p_B(v)$ is a penalty function to avoid blocks that are heavily overloaded. To be more precise, after some experiments, we fixed the penalty function to $p_B(v) = 2^{\max(0,100\frac{c(B)+c(v)}{L_{\max}})}$, where $L_{\max}$ is the upper bound for the block weight. Note that slight imbalances (e.g., overloaded blocks) can usually be fixed by the refinement algorithms implemented within KaFFPa. For matching-based coarsening, the uncoarsening is straightforward: a vertex is assigned to the block of the corresponding coarse vertex.

Karlsruhe Fast Flow Partitioner. Since we integrated different coarsening schemes into the multilevel graph partitioner KaFFPa [Sanders and Schulz 2011], we now briefly outline the techniques used by KaFFPa during uncoarsening. After a matching is uncontracted, local search-based refinement algorithms move nodes between block boundaries to reduce the cut while maintaining the balance constraint. Local improvement algorithms are usually variants of the FM algorithm [Fiduccia and Mattheyses 1982]. The variant used by KaFFPa is organized in rounds. In each round, a priority queue P is used that is initialized with all vertices that are incident to more than one block in a random order. The priority is based on the gain $g(i) = \max_P g_P(i)$, where $g_P(i)$ is the decrease in edge cut when moving i to block P. Local search then repeatedly looks for the highest gain node v and moves it to the corresponding block that maximizes the gain. However, in each round, a node is moved at most once. After a node is moved, its unmoved neighbors become eligible—that is, its unmoved neighbors are inserted into the priority queue. When a stopping criterion is reached, all movements after the best-found cut that occurred within the balance constraint are undone. This process is repeated several times until no improvement is found.

Max-flow min-cut local improvement. During the uncoarsening phase, KaFFPa additionally uses more advanced refinement algorithms. The first method is based on max-flow min-cut computations between pairs of blocks—in other words, a method to improve a given bipartition. Roughly speaking, this improvement method is applied between all pairs of blocks that share a nonempty boundary. The algorithm basically constructs a flow problem by growing an area around the given boundary vertices of a pair of blocks such that each *s*-*t* cut in this area yields a feasible bipartition of the original graph/pair of blocks *within* the balance constraint. One can then apply a max-flow min-cut algorithm to obtain a min-cut in this area and therefore a nondecreased cut between the original pair of blocks. This can be improved in multiple ways, such as by iteratively applying the method, searching in larger areas for feasible cuts, and applying most balanced minimum cut heuristics. For more details, we refer the reader to Sanders and Schulz [2011].

Multitry FM. The second method for improving a given partition is called *multitry FM*. This local improvement method moves nodes between blocks to decrease the cut. Previous k-way methods were initialized with *all* boundary nodes—in other words, all boundary nodes were eligible for movement at the beginning. Roughly speaking, the multitry FM algorithm is a k-way improvement method that is initialized with a *single* boundary node, thus achieving a more localized search. This is repeated several rounds. For more details about the multitry FM algorithm, we refer the reader to Sanders and Schulz [2011].

Table I. Description of the Six Configurations Used for the Computational Experiments

ECO	Represents the classical KaFFPaEco configuration, a good trade-off of partition quality and runtime.
ECO-ALG	Same refinement as in ECO; coarsening uses the GPA algorithm at each level, and the edge rating function employs algebraic distances—in other words, it uses the rating function $\text{ex\_alg}(e) := \text{expansion}^{*2}(e)/\rho_e$.
F-CYCLE	Represents the classical KaFFPaStrong configuration using strong refinement schemes and the F-cycle scheme, with the purpose of achieving high partition quality; this configuration achieved the best-known partitions for many instances from Benchmark I in 2010 [Sanders and Schulz 2011].
STRONG	Uses the same refinement and matching schemes as in the F-CYCLE configuration; however, here only one single V-cycle is performed.
AMG-ECO	AMG coarsening based on algebraic distances with interpolation order at most 2 and refinement as in ECO.
AMG	Same coarsening as in AMG-ECO and same refinement as in STRONG.

4. EXPERIMENTAL EVALUATION

The AMG coarsening was implemented separately based on the coarsening for linear ordering solvers from Ron et al. [2011] and was integrated into KaFFPa [Sanders and Schulz 2011]. The computational experiments have been performed with six configurations of KaFFPa, which are presented in Table I. All configurations use the described FM algorithm and flows for the refinement. The strong configurations further employ flows using larger areas, multitry FM, and F-cycles. A detailed description of the refinement configurations can be found in Sanders and Schulz [2011]. Throughout this section, because of the respective similar running times, we concentrate on two groups of comparison: for fast versions (AMG-ECO, ECO, ECO-ALG) and for strong versions (AMG, STRONG, F-CYCLE). To be precise, usually the running time of F-CYCLE is bigger than that of STRONG and AMG. However, the running time gap between fast and strong versions is even more significant on average. Since the main goal of this article is to introduce the AMG coarsening with different uncoarsening configurations, most of the comparisons will be of type AMG versus respective non-AMG ratios. A comprehensive comparison of the F-CYCLE and the STRONG configuration can be found in Sanders and Schulz [2011].

All experiments are performed with a fixed imbalance factor of 3%. We also checked other small values, namely, 0%, 1%, and 5%; however, no significant difference in the comparison of the respective methods was observed.

4.1. Benchmark I: Walshaw's Partitioning Archive

Chris Walshaw's benchmark archive [Soper et al. 2004] is a collection of real-world instances for the partitioning problem. The rules used there imply that the running time is not an issue, but one wants to achieve minimal cut values for $k \in \{2, 4, 8, 16, 32, 64\}$ and balance parameters $\epsilon \in \{0, 0.01, 0.03, 0.05\}$. It is the most used graph partitioning benchmark in the literature. The main properties of the 34 graphs are shown later in Table V (see Appendix B). Most of the graphs of the benchmark come from finite-element applications; however, there are also some graphs from VLSI design and a road network. Over the years, many different heuristics have been tested and adjusted on this benchmark, so many heuristics are able to obtain good results on these graphs.

In Figure 3 , we present the results of the comparison of the algorithms on these graphs for different numbers of blocks k. The horizontal axes represent ordered graphs from the benchmark (however, the ordering itself will be different for each curve). The vertical axes are for ratios that represent the comparison of averages of final results for a pair of methods. Each figure contains four curves. Each curve corresponds to a comparison of the following pairs of methods: ECO versus AMG-ECO, ECO-ALG

Table II. Computational Comparison for Benchmark I

k	ECO/ECO-ALG	ECO-ALG/ECO-AMG	STRONG/AMG	F-CYCLE/AMG
2	1.026	1.034	1.013	1.012
4	1.053	1.021	1.009	1.004
8	1.019	1.023	0.998	0.995
16	1.015	1.012	1.001	0.999
32	1.008	1.017	1.003	1.002
64	1.004	1.009	1.000	0.997

Note: Each number corresponds to the ratio of averages of final cuts for pair of methods in the column title and k given in the row.

Table III. Computational Comparison for Scale-Free Graphs

k	$\frac{\text{ECO}}{\text{ECO-ALG}}$ quality	$\frac{\text{ECO}}{\text{ECO-ALG}}$ full time	$\frac{\text{ECO}}{\text{ECO-ALG}}$ uncoarsening time	$\frac{\text{ECO-ALG}}{\text{AMG-ECO}}$ quality	$\frac{\text{ECO-ALG}}{\text{AMG-ECO}}$ uncoarsening time
2	1.38	0.77	1.62	1.16	3.62
4	1.24	1.32	1.85	1.11	2.14
8	1.15	1.29	1.45	1.07	1.94
16	1.09	1.27	1.33	1.06	1.69
32	1.06	1.18	1.23	1.00	1.60
64	1.06	1.13	1.13	1.01	2.99

versus AMG-ECO, STRONG versus AMG, and F-CYCLE versus AMG. Each point on the curves corresponds to the ratio between the average over 10 runs of one method and the average over 10 runs of another method. Each run depends on different random seeds and thus can produce different results. For example, the last point at the black solid curve in Figure 3(a) has a value of 2.03, which means that

$$\frac{\text{average(ECO final cut given seed } s_1, \ldots, \text{ ECO final cut given seed } s_{10})}{\text{average(AMG-ECO final cut given seed } s_1, \ldots, \text{ AMG-ECO final cut given seed } s_{10})} = 2.03$$

in experimental series for $k = 2$. A comparison of the running time for uncoarsening phases is presented in Figure 4. Each point on the curves in Figure 4 corresponds to a ratio of uncoarsening running times of two methods. We observed that the uncoarsening performances of fast versions (ECO, ECO-ALG, AMG-ECO) are more or less similar to each other. The uncoarsening of a STRONG V-cycle is somewhat slower than AMG because of the density of coarse levels. With regard to partition quality, we observed that there is not much difference between all of the methods here. The averages are summarized in Table II. Full results are summarized in Safro et al. [n.d.].

4.2. Benchmark II: Scale-Free Networks

In scale-free networks, the distribution of vertex degrees asymptotically follows the power-law distribution. Examples of such networks include WWW links, social communities, and biological networks. These types of networks often contain irregular parts and long-range links (in contrast to Benchmark I) that can confuse both contraction and AMG coarsening schemes. Since Walshaw's benchmark does not contain graphs derived from such networks, we evaluate our algorithms on 15 graphs collected from Bader et al. [n.d.] and Lescovec [n.d.]. Table VI (presented later) summarizes the main properties of these graphs. Full information about these graphs, along with the computational results, is available at Safro et al. [n.d.].

The results of the comparison on scale-free graphs are presented in Figure 5. Because of the large running time of the strong configurations on these graphs, we compare only the fast versions of AMG and matching-based coarsening. Each figure corresponds to

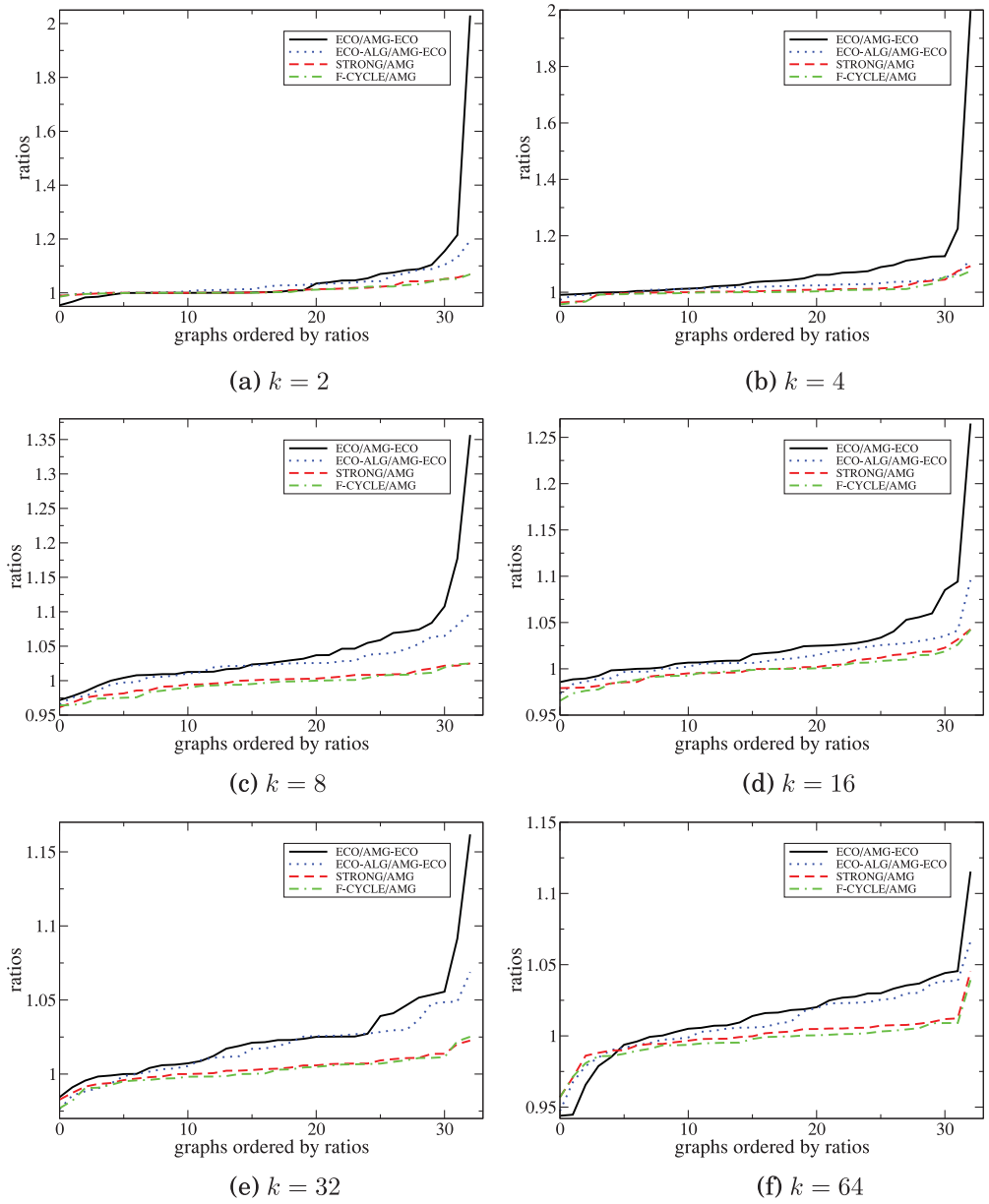


Fig. 3. Comparison of coarsening schemes on Walshaw's benchmark graphs. Parts (a) through (f) contain results of comparison for $k = 2$, 4, 8, 16, 32, and 64, respectively. Each figure contains four curves that correspond to ECO/AMG-ECO, ECO-ALG/AMG-ECO, STRONG/AMG, and F-CYCLE/AMG ratios, respectively. Each point on a curve corresponds to the average of ratios of final cuts related to one graph.

a different number of blocks k. The horizontal axes represent graphs from the benchmark. The vertical axes are for ratios that represent comparison of averages of final results for a pair of methods. Each graph corresponds to one quadruple of bars. First, second, third, and fourth bars represent averages of ratios ECO/AMG-ECO, ECO-ALG/AMG-ECO after finest refinement, ECO/AMG-ECO, and ECO-ALG/AMG-ECO before

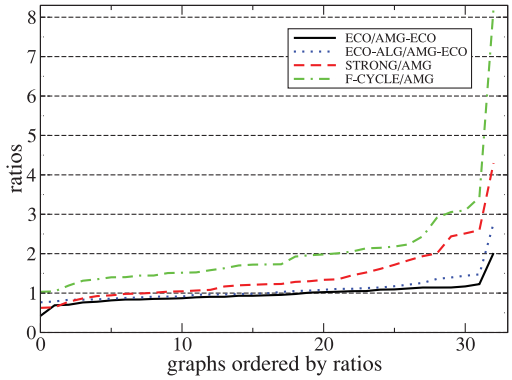


Fig. 4. Comparison of uncoarsening running time on Walshaw's benchmark graphs for $k = 32$. The figure contains four curves that correspond to ECO/AMG-ECO, ECO-ALG/AMG-ECO, STRONG/AMG, and F-CYCLE/AMG ratios, respectively. Each point on the curves corresponds to the average of ratios of final cuts related to one graph.

finest refinement, respectively. As in the previous case, the averages are calculated over 10 runs.

4.3. Benchmark III: Potentially Hard Graphs for Fast *k*-Partitioning Algorithms

Today, multilevel strategies represent one of the most effective and efficient *generic* frameworks for solving the graph partitioning problem on large-scale graphs. The reason is obvious: given a successful global optimization technique X for this problem, one can consider applying it locally by introducing a chain of subproblems along with fixed boundary conditions. Given this and if the coarsening preserves the structural properties of the graph well enough, the multilevel heuristic can behave better and work faster than a direct global application of optimization technique X. Examples of such combinations include FM/KL, spectral, and min-cut/max-flow techniques with multilevel frameworks. When can the multilevel framework produce low-quality results?

We present a simple strategy for checking the quality of multilevel schemes. To construct a potentially hard instance for gradual multilevel projections, we consider a mixture of graphs that are weakly connected with each other. These graphs have to possess different structural properties (such as finite-element faces, power-law degree distribution, and density) to ensure nonuniform coarsening and mutual aggregation of well-separated graph regions. Such mixtures of structures may have a twofold effect. First, they can force the algorithm to contract incorrect edges; second, they can attract a "too strong" refinement to reach a local optimum, which can contradict better local optimums at finer levels. The last situation has been observed in different variations as well in multilevel linear ordering algorithms [Safro et al. 2008]. In other words, the uneven aggregation with respect to the scales (not to be confused with uneven sizes of clusters) can lead refinement algorithms to wrong local attraction basins. Examples of graphs that contain such mixtures of structures include multimode networks [Tang et al. 2008] and logistics multistage system networks [Stock 2006]. In general, such graphs can be difficult not only to the multilevel algorithms.

We created a benchmark (available at Bader et al. [2013]) with potentially hard mixtures. Each graph in this benchmark represents a star-like structure of different subgraphs $S_0, \ldots, S_t$. Graphs $S_1, \ldots, S_t$ are weakly connected to the center S_0 by random edges. Since all constituent subgraphs are sparse, a faster aggregation of them has been achieved by adding more than one random edge to each boundary node.

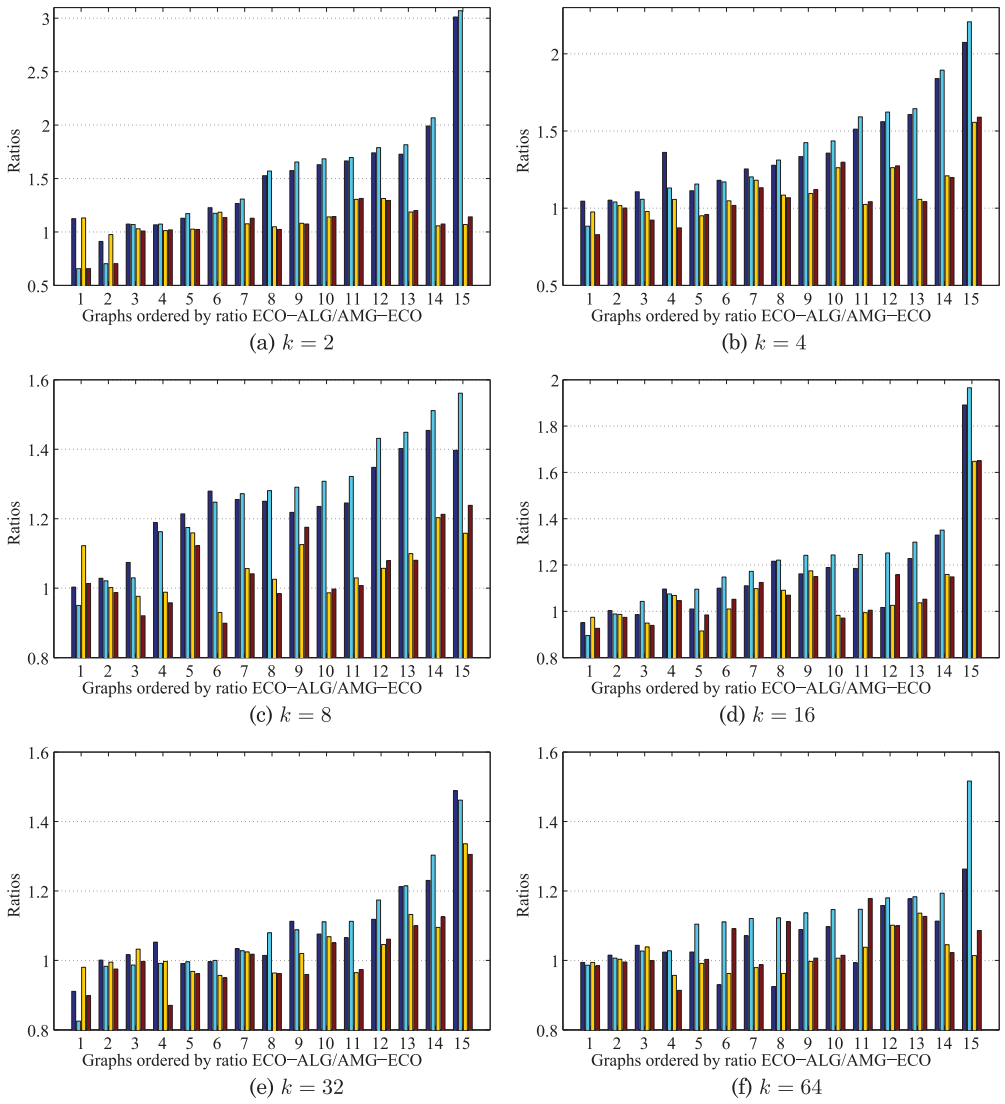


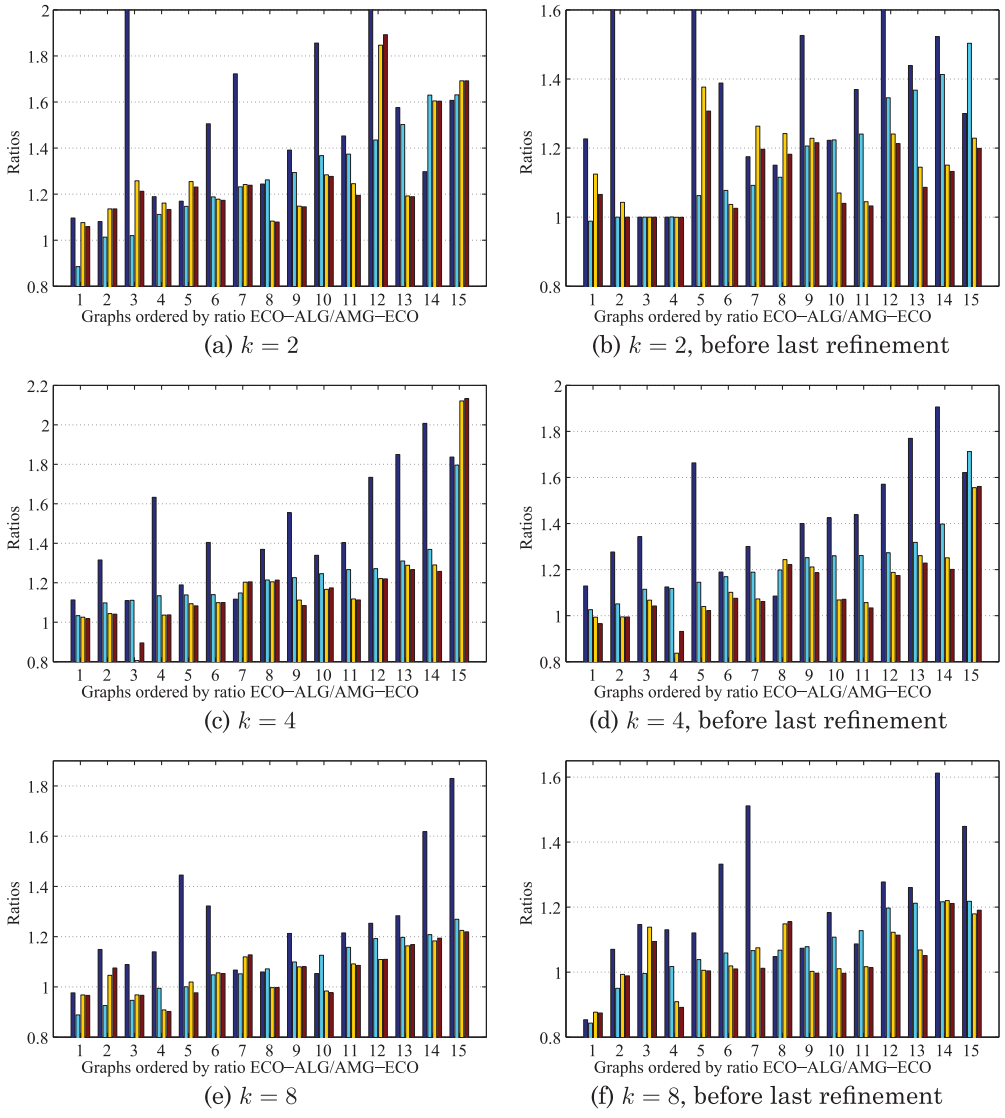
Fig. 5. Comparison of coarsening schemes on scale-free graphs. Parts (a) through (f) contain results of comparison for $k = 2, 4, 8, 16, 32$, and 64, respectively. Each quadruple of bars corresponds to one graph. First, second, third, and fourth bars represent averages of ratios (of final cuts) ECO/AMG-ECO, ECO-ALG/AMG-ECO after refinement, ECO/AMG-ECO, and ECO-ALG/AMG-ECO before refinement, respectively. Three exceptionally high ratios on both figures are between 2.1 and 3.

The total number of edges between each S_i and S_0 was less than 3% out of the total number of edges in S_i. We considered the mixtures of the following structures: social networks, finite-element graphs, VLSI chips, peer-to-peer networks, and matrices from optimization solvers. The main properties of the graphs are shown later in Table VII (see Appendix B).

The comparison on this benchmark is demonstrated in Figure 6. Each graph corresponds to one quadruple of bars. The first, second, third, and the fourth bar represent

Table IV. Computational Comparison for Potentially Hard Graphs

k	ECO / ECO-ALG quality	ECO / ECO-ALG full time	ECO-ALG / AMG-ECO quality	ECO-ALG / AMG-ECO uncoarsening time	STRONG / AMG quality	STRONG / AMG uncoarsening time	F-CYCLE / AMG quality
2	1.42	0.51	1.18	0.55	1.15	2.11	1.11
4	1.15	0.88	1.23	0.64	1.13	1.69	1.12
8	1.12	1.08	1.08	0.98	1.05	1.37	1.04



(a) $k = 2$ (b) $k = 2$, before last refinement

(c) $k = 4$ (d) $k = 4$, before last refinement

(e) $k = 8$ (f) $k = 8$, before last refinement

Fig. 6. Comparison of coarsening schemes on hard examples. Parts (a), (c), and (e) contain results (averages of cuts' ratios) of comparison before applying finest-level refinement. Parts (b), (d), and (f) contain results of comparison of final results (averages of final cuts' ratios). Each quadruple of bars corresponds to one graph. First, second, third, and fourth bars represent averages of ratios ECO/AMG-ECO, ECO-ALG/AMG-ECO, STRONG/AMG, and F-CYCLE/AMG, respectively. Four exceptionally high ratios on both figures are between 3.5 and 5.7.

averages over 10 ratios of ECO/AMG-ECO, ECO-ALG/AMG-ECO, STRONG/AMG, and F-CYCLE/AMG, respectively. In almost all experiments, we observed that introduction of algebraic distance as a measure of connectivity plays a crucial role in both fast versions AMG-ECO and ECO-ALG, as it helps to separate the subgraphs and postpone their aggregation into one mixture. We also observe that both fast and slow AMG coarsenings almost always lead to better results. Note that in contrast to Benchmarks I and II, the uncoarsening of ECO-ALG is significantly faster than that of AMG-ECO.

4.4. Discussion

Role of the algebraic distance. In this work, the importance of the algebraic distance as a measure of connectivity strength for graph partitioning algorithms has been justified in almost all experimental settings. In particular, the most significant gap was observed between ECO and ECO-ALG versions (see all benchmarks), which confirms preliminary experiments in Chen and Safro [2011], where the algebraic distance has been used at the finest level only. The price for improvement in the quality is the additional running time for Jacobi overrelaxation, which can be implemented by using the most suitable (parallel) matrix-vector multiplication method. However, in cases of strong configurations and/or large irregular instances, the difference in the running time becomes less influential, as it is not comparable to the amount of work in the refinement phase. For example, for the largest graph in Benchmark I (auto, $|V| = 448695$, $|E| = 3314611$), the ECO coarsening is approximately 10 times faster than that in the ECO-ALG; however, for both configurations when $k = 64$, it takes less than 3% of the total time. Note that for irregular instances from Benchmark II, already starting $k = 4$, the total running time for ECO becomes bigger than for ECO-ALG (see Table III). More examples of trade-off between changes in the objectives and those in the running times on Benchmark III are presented in Figure 7.

Does AMG coarsening help? The positive answer to this question is given mostly by Benchmarks II and III, which contain relatively complex instances (Tables III and IV). On Benchmark III, we have demonstrated that the AMG configuration is superior to F-CYCLE, which runs significantly longer. This result is in contrast to Benchmark I, in which we did not observe any particular class of graphs that corresponded to stable significant difference in favor of one of the methods in pairs ECO-ALG versus AMG-ECO and STRONG versus AMG. However, we note that in both Benchmarks I and II, several graphs exhibited that AMG versions yield to the respective matching for large k. The problem is eliminated when we stabilize ρ by using more relaxations according to Theorem 4.2 in Ron et al. [2011]. We cannot present here the exact comparison of coarsening running times because their underlying implementations are very different. Theoretically, however, if in both matching and AMG configurations the algebraic distance is used and when the order of interpolation in AMG is limited by 2 (and usually it is 1, meaning that the coarse graphs are not dense like in Chevalier and Safro [2009]), the exact complexity of AMG coarsening is not supposed to be bigger than that of matching.

5. CONCLUSIONS

We introduced a new coarsening scheme for multilevel graph partitioning based on AMG coarsening. One of its most important components—namely, the algebraic distance connectivity measure—has been incorporated into the matching coarsening schemes. Both coarsening schemes have been compared under fast and strong configurations of refinement. In addition to known benchmarks, we introduced new potentially hard graphs for large-scale graph partitioning solvers (available at Bader

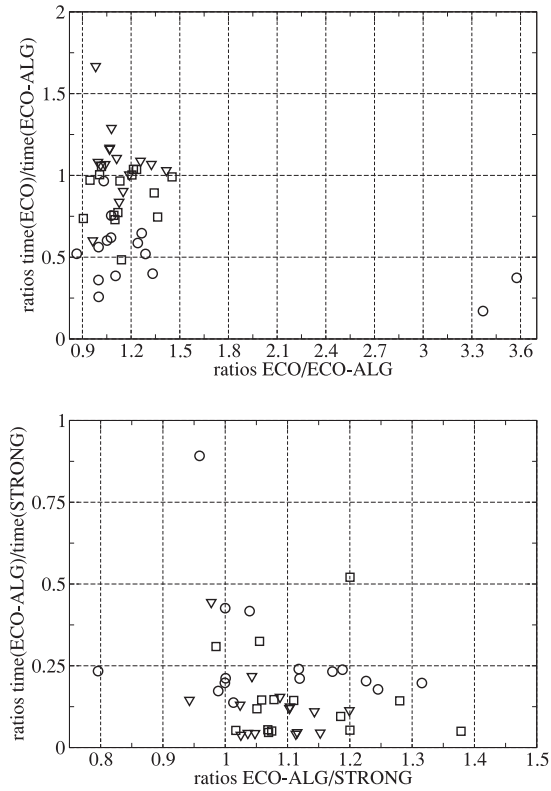


Fig. 7. Benchmark III. Trade-off between changes in the objectives (horizontal axis) and those in the running times (vertical axis) on Benchmark III. Data points for $k = 2$, 4, and 8 are represented by circles, squares, and triangles, respectively. Average ratios (of both time and final cuts) are calculated each over 10 runs, similarly to previous figures. The top and bottom figures describe the comparison for ECO versus ECO-ALG and ECO-ALG versus STRONG configurations, respectively.

et al. [2013]). As the main conclusion of this work, we emphasize the success of the proposed AMG coarsening and the algebraic distance connectivity measure between nodes demonstrated on highly irregular instances. One has to take into account the trade-off between increased running time when using algebraic distance and improved quality of the partitions. The increasing running time becomes less tangible with growth of graph size compared with the complexity of the refinement phase.

Many opportunities remain to improve the coarsening schemes for graph partitioning. We demonstrated the crucial importance of the connectivity strength metrics (especially for fast versions of the algorithm), which raises the question of how one can use these metrics at the uncoarsening phase. Preliminary experiments show that this has the potential to improve fast versions even more. Another issue that requires more insight is related to the balancing of AMG aggregates. We observed a number of examples for which the unbalanced coarsening produces noticeably better results.

APPENDIX A: BRIEF BACKGROUND ON AMG

For comprehensive surveys on multigrid methods and stationary iterative relaxations, we refer the reader to Brandt and Ron [2003] and Heath [2002], respectively. AMG methods were originally developed for solving linear systems of equations that describe

the discretization of partial differential equations; up to now, these methods represent the most effective class of numerical algorithms for them. The common approach of all multigrid algorithms (not only AMG) is to construct a sequence of representations of the original problem at increasingly larger (coarser) scales and to combine local processing (relaxation) at each scale with various interscale interactions. Coarse-scale equations are typically dictated by the fine equations, whereas large-scale corrections for the solutions of fine scales are supplied by coarse-scale solutions. Large-scale changes are effectively calculated on coarse grids, based on information previously gathered from finer discretizations. During the past three decades, multigrid methods were adapted and generalized for many computational tasks in various disciplines. Multigrid methods are known to be scalable and efficient because they can solve a system with only linear time and space complexity. Moreover, the nature of these algorithms allows relatively easy distribution of the main parts of the task among parallel machines, which is what makes these methods ideal for solving large-scale computational problems.

In AMG, the goal is to solve a system of equations

$$Ax = b \left(\text{or equivalently to minimize } \frac{1}{2}x^T Ax - b^T x\right), \tag{4}$$

where $A \in \mathbb{R}^{n\times n}$ is symmetric positive definite[2] and $x, b \in \mathbb{R}^n$. One way to solve such a system is by using iterative methods that begin with an initial guess and successively improve it until the solution is accurate enough. Examples of such methods (called *relaxations*) include Jacobi, Gauss-Seidel, and successive overrelaxation methods that can be written as

$$\tilde{x}^{(k+1)} = T\tilde{x}^{(k)} + v, \tag{5}$$

where $\tilde{x}^{(k)}$ is an approximate solution at the kth iteration and T is an iterator usually obtained by splitting A. The splitting will be different for different types of relaxations. The relaxation is of interest if it converges to the optimal solution of (4). However, for most of the existing relaxations and linear systems that can be solved by them, the convergence is typically slow. A common feature of such relaxations is that at each step, corrections to $\tilde{x}^{(k)}$ are calculated based on a small number of unknowns—in other words, the relaxation is a local process that after a small number of iterations smooths the remaining error.

The main idea of the multigrid is to exploit the fact that a smooth (relaxed) error can be approximated by using fewer variables than the original error. Following a small number of relaxation iterations, the remaining smooth error e and also the solution $\tilde{x}$ can be approximated by a coarser system with fewer new variables. The main issue in any coarsening scheme is how to define the coarse variables, the operator of restriction that describes a problem by using fewer variables, and the operator of interpolation that projects the solution of the coarse problem onto fine scale. In AMG, the set of coarse variables C is chosen as a subset of the set of fine variables F, or, more generally, each coarse variable is compounded of fractions of a small number of fine variables. This aggregation process is performed around each fine variable (which entirely contributes itself to the respective aggregate) chosen into the set C. This set must be constructed so that all other fine variables in $F \setminus C$ will be strongly coupled to C. The notion of strong coupling varies in different AMG methods. The main idea of a strong coupling is to ensure that the variable x_i need not be in C if it is already connected to those variables

[2]We consider symmetric positive definite for simplicity. In general, the approach can be extended for other types of matrices.

whose solution can be interpolated well enough to x_i. The classical criterion for this is Equation (6). During the past two decades, many advanced methods to measure the strength of coupling were proposed. Algebraic distance also belongs to this class of measures. In fact, for current $\tilde{x}$, the problem in (4) will be solved by reformulating it as a residual problem $Ae = r$, where $e = x - \tilde{x}$ is an error and $r = b - A\tilde{x}$ is a residual. One of the classical multigrid algorithms, called *correction scheme*, is presented in Algorithm 2.

ALGORITHM 2: Correction scheme algorithm MG

input: A, $\tilde{x}$, b

if A *is small enough* **then**
  Solve $A\tilde{x} = b$;
else
  Apply error smoothing (relaxation) sweeps (5);
  Define A^c, $e^c = 0$ and $r^c = \uparrow_f^c r$;
  $MG(A^c, e^c, r^c)$;
  Correct $\tilde{x} = \tilde{x} + \uparrow_c^f e^c$;
  Apply error smoothing (relaxation) sweeps (5);

In this algorithm, A^c, e^c, and r^c are the coarse matrix, the coarse error, and the coarse residual, respectively. In AMG, the main part of the procedure for choosing the set of coarse variables C contains a sweep through all fine variables transferring to C those that are still not strongly connected to already chosen variables. The strength of the connection between a currently visited variable in F and its seeds (those that are already in C) is controlled by the parameter $Q \in (0, 1]$. In particular, $i \in F$ is strongly connected to C if

$$\frac{\sum_{j \in C} a_{ij}}{\sum_{j \in F} a_{ij}} \geq Q. \tag{6}$$

One of the most traditional approaches for derivation of the coarse equations A^c in AMG is the Galerkin operator

$$A^c = \uparrow_f^c A \uparrow_c^f, \tag{7}$$

which projects the fine system of equations to the coarser scale. Usually, for symmetric and positive definite matrices A, the restriction mapping is $\uparrow_f^c = (\uparrow_c^f)^T$. The (i, J)th entry of $\uparrow_c^f$ represents the strength of the connection between fine variable i and coarse variable J and $e = \uparrow_c^f e_c$. The entries of $\uparrow_c^f$ are referred to as interpolation weights, and they describe both the coarse-to-fine and fine-to-coarse variable relations. To control the complexity of the coarse-scale system (the number of nonzero elements in A^c), the number of fractions into which a fine variable can be divided (and thus attached to the coarse variables) is bounded by the order of interpolation. If a row in $\uparrow_f^c$ contains too many nonzero elements, then it is likely to increase the number of nonzeros in A^c. Thus, this number is usually controlled by different approaches that measure the strength of connectivity (or importance) between fine and coarse variables. The algebraic distance that we use in the coarsening (see Section 3.1) belongs to this class of approaches.

In our context, the error smoothing relaxation is represented by the algebraic distance and the graph Laplacian is coarsened in the spirit of the presented AMG scheme.

APPENDIX B: BASIC PROPERTIES AND BASELINE RESULTS OF BENCHMARK INSTANCES USED

In Tables VII, IX, and X we present computational results for Benchmarks 2, 3, and 1, respectively.

Table V. Basic Properties of Walshaw Benchmark Graphs

Graph	n	m
add20	2395	7462
data	2851	15093
3elt	4720	13722
uk	4824	6837
add32	4960	9462
bcsstk33	8738	291583
whitaker3	9800	28989
crack	10240	30380
wing_nodal	10937	75488
fe_4elt2	11143	32818
vibrobox	12328	165250
bcsstk29	13992	302748
4elt	15606	45878
fe_sphere	16386	49152
cti	16840	48232
memplus	17758	54196
cs4	22499	43858
bcsstk30	28924	1007284
bcsstk31	35588	572914
fe_pwt	36519	144794
bcsstk32	44609	985046
fe_body	45087	163734
t60k	60005	89440
wing	62032	121544
brack2	62631	366559
finan512	74752	261120
fe_tooth	78136	452591
fe_rotor	99617	662431
598a	110971	741934
fe_ocean	143437	409593
144	144649	1074393
wave	156317	1059331
m14b	214765	1679018
auto	448695	3314611

Table VI. Basic Properties of Scale-Free Graphs

Graph	n	m
as-22july06	22963	48436
as-skitter	554930	5797663
citationCiteseer	268495	1156647
coAuthorsCiteseer	227320	814134
coAuthorsDBLP	299067	977676
coPapersDBLP	540486	15245729
email-EuAll	16805	60260
web-Google	356648	2093324
wiki-Talk	232314	1458806
coPapersCiteseer	434102	16036720
loc-brightkite	56739	212945
loc-gowalla	196591	950327
p2p-Gnutella04	6405	29215
PGPgiantcompo	10680	24316
soc-Slashdot0902	28550	379445

Table VII. Basic Properties of Potentially Hard Graphs

Graph	n	m
barth5_1Ksep_50in_5Kout	32212	101805
bcsstk30_500sep_10in_1Kout	58348	2016578
befref_fxm_2_4_air02	14109	98224
bump2_e18_aa01_model1_crew1	56438	300801
c-30_data_data	11023	2184
c-60_data_cti_cs4	85830	241080
data_and_seymourl	9167	55866
finan512_scagr7-2c_rlfddd	139752	552020
mod2_pgp2_slptsk	101364	389368
msc10848_300sep_100in_1Kout	21996	1221028
sctap1-2b_and_seymourl	40174	140831
south31_slptsk	39668	189914
vibrobox_scagr7-2c_rlfddd	77328	435586
p0291_seymourl_iiasa	10498	53868
model1_crew1_cr42_south31	45101	189976

Table VIII. Best Baseline Results (Final Cuts, AMG-ECO) for Scale-Free Graphs

Graph	$k=2$	$k=4$	$k=8$	$k=16$	$k=32$	$k=64$
as-22july06	3525	7793	11816	14417	18846	21187
as-skitter	233880	426555	631860	922479	1275580	1723340
citationCiteseer	33019	69441	105597	143682	190887	231361
coAuthorsCiteseer	19119	35488	49720	58317	67397	74770
coAuthorsDBLP	49268	82139	105319	124426	145250	161705
coPapersDBLP	486768	844223	1184020	1505970	1821300	2058860
email-EuAll	682	5606	13725	22457	27140	35176
loc-brightkite_edges	20451	36086	47454	55500	65742	75531
loc-gowalla_edges	56119	132232	191548	253907	317408	352958
p2p-Gnutella04	7516	12335	15288	17515	20191	19543
PGPgiantcompo	378	742	1159	1842	2667	3123
soc-Slashdot0902	98953	169422	248887	286205	314022	330046
web-Google	12578	20243	24031	26934	30826	48136
wiki-Talk	76890	253898	497122	835821	982291	1092920
coPapersCiteseer	266289	507025	712150	877103	1041120	1166210

Table IX. Best Baseline Results (Final Cuts) for Potentially Hard Graphs

	$k=2$	$k=2$	$k=4$	$k=4$	$k=8$	$k=8$
Graph	AMG	AMG-ECO	AMG	AMG-ECO	AMG	AMG-ECO
barth5_1Ksep_50in_5Kout	3735	3735	6102	6104	7765	7760
bcsstk30_500sep_10in_1Kout	617	617	13562	13572	35335	35778
befref_fxm_2_4_air02	3638	3638	28948	29027	45856	46145
bump2_e18_aa01_model1_crew1	29701	29658	63472	64374	85619	86947
c-30_data_data	1512	1673	5520	5199	12086	15264
c-60_data_cti_cs4	4405	4425	7332	7420	9960	10153
data_and_seymourl	7185	7180	14931	14675	19299	19714
finan512_scagr7-2c_rlfddd	13573	13603	43597	43877	61776	63678
mod2_pgp2_slptsk	29625	32122	60061	65451	84268	89613
model1_crew1_cr42_south31	24907	27359	47397	48893	68135	70165
msc10848_300sep_100in_1Kout	737	737	32485	32791	72497	74941
p0291_seymourl_iiasa	6708	6688	15779	15925	21527	21707
sctap1-2b_and_seymourl	16036	20466	26367	26882	37045	38057
south31_slptsk	24425	29005	43724	44129	54376	55303
vibrobox_scagr7-2c_rlfddd	35450	35834	54662	54905	78637	79762

Table X. Best Bestline Results (Final Cuts) for Walshaw's Benchmark

	$k=2$		$k=4$		$k=8$		$k=16$		$k=32$		$k=64$	
graph	AMG	AMG-ECO	AMG	AMG-ECO	AMG	AMG-ECO	AMG	AMG-ECO	AMG	AMG-ECO	AMG	AMG-ECO
3elt	87	87	201	203	343	348	570	585	986	1009	1606	1640
4elt	137	138	319	321	525	537	954	964	1574	1609	2610	2674
598a	2367	2367	7885	7930	16173	16507	25599	26206	39195	40696	57438	58696
add20	675	668	1240	1238	1752	1761	2147	2208	2632	2807	3149	3149
add32	10	10	33	33	66	66	117	117	212	212	512	515
bcsstk29	2818	2818	8031	8144	14113	14262	22890	23412	35640	35959	57195	58076
bcsstk30	6251	6251	16497	16685	34302	34371	70998	72006	116243	118039	175079	177395
bcsstk31	2676	2676	7314	7418	13284	13335	23922	24135	38267	38895	59650	61037
bcsstk32	4938	4938	8749	9021	20846	21713	36309	37020	60336	62177	93654	94932
bcsstk33	10064	10064	21355	21555	34990	35433	55513	55775	79610	81416	110598	111552
brack2	684	706	2836	2856	6961	7317	11548	11959	17600	18178	26218	26983
crack	183	186	360	374	688	699	1121	1138	1721	1756	2606	2676
cs4	365	377	954	983	1487	1543	2146	2221	2975	3102	4112	4289
cti	352	352	928	945	1716	1788	2776	2946	4120	4346	5988	6318
data	193	196	390	390	677	684	1162	1156	1882	1924	2949	2944
fe_4elt2	130	130	349	349	609	616	1003	1022	1636	1668	2557	2593
fe_body	262	262	608	609	1064	1083	1791	1805	2901	3008	4888	5037
fe_ocean	311	311	1720	1800	3963	4102	7951	8375	12753	13852	20368	21508
fe_pwt	340	345	702	700	1442	1445	2805	2815	5572	5587	8322	8503
fe_rotor	1959	1959	7471	7591	12975	13057	20533	20988	31785	32310	46903	47888
fe_sphere	384	384	786	806	1181	1234	1747	1844	2540	2688	3638	3908
fe_tooth	3795	3906	6871	7064	11484	11702	17599	18352	25612	26113	35286	36296
finan512	162	162	324	324	648	648	1296	1296	2592	2592	10652	10839
m14b	3823	3823	13016	13102	25505	26053	43110	44236	66211	67767	97831	100471
memplus	5714	5701	10290	10540	12293	12436	14054	14225	15320	15972	16689	17643
t60k	73	77	207	208	464	471	870	885	1385	1418	2175	2247
uk	18	19	40	42	83	84	151	157	265	267	431	435
wave	8641	8676	16787	16860	28606	29401	41628	44043	61903	62909	84763	87200
whitaker3	126	128	378	384	663	661	1118	1141	1702	1735	2600	2712
wing	776	800	1698	1739	2531	2611	3992	4283	5701	5969	7908	8603
wing_nodal	1694	1723	3597	3610	5511	5577	8312	8251	11993	12197	16335	16716
144	6472	6529	15592	15836	25639	25928	38973	40103	56582	57748	79895	83005
auto	9685	9685	25961	26107	44977	45454	76303	78190	120734	122969	173752	181609

REFERENCES

D. A. Bader, H. Meyerhenke, P. Sanders, and D. Wagner. 2013. 10th DIMACS Implementation Challenge—Graph Partitioning and Graph Clustering. Retrieved October 20, 2014, from http://www.cc.gatech.edu/dimacs10/.

G. Bartel, C. Gutwenger, K. Klein, and P. Mutzel. 2010. An experimental evaluation of multilevel layout methods. In *Graph Drawing. Lecture Notes in Computer Science*, Vol. 6502. Springer, 80–91.

A. Brandt. 2001. Multiscale scientific computation: Review 2001. In *Multiscale and Multiresolution Methods.* Lecture Notes in Computational Science and Engineering, Vol. 20. Springer, 3–95.

A. Brandt and D. Ron. 2003. Multigrid solvers and multilevel optimization strategies. In *Multilevel Optimization in VLSICAD*. Combinatorial Optimization, Vol. 14. Springer, 1–69.

T. N. Bui and C. Jones. 1992. Finding good approximate vertex and edge partitions is NP-hard. *Information Processing Letters* 42, 3, 153–159.

T. N. Bui and B. R. Moon. 1996. Genetic algorithm and graph partitioning. *IEEE Transactions on Computers* 45, 7, 841–855. DOI:http://dx.doi.org/10.1109/12.508322

J. Chen and I. Safro. 2011. Algebraic distance on graphs. *SIAM Journal on Scientific Computing* 33, 6, 3468–3490.

C. Chevalier and I. Safro. 2009. Comparison of coarsening schemes for multilevel graph partitioning. In *LION*. Lecture Notes in Computer Science, Vol. 5851. Springer, 191–205.

I. Dhillon. 2005. A fast kernel-based multilevel algorithm for graph clustering. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 629–634.

D. Drake and S. Hougardy. 2003. A simple approximation algorithm for the weighted matching problem. *Information Processing Letters* 85, 211–213.

N. Fan and P. M. Pardalos. 2010. Linear and quadratic programming approaches for the general graph partitioning problem. *Journal of Global Optimization* 48, 1, 57–71. DOI:http://dx.doi.org/10.1007/s10898-009-9520-1

C. M. Fiduccia and R. M. Mattheyses. 1982. A linear-time heuristic for improving network partitions. In *Proceedings of the 19th Conference on Design Automation*. 175–181.

P. O. Fjallstrom. 1998. Algorithms for graph partitioning: A survey. *Linkoping Electronic Articles in Computer and Information Science* 3, 10.

W. W. Hager and Y. Krylyuk. 1999. Graph partitioning and continuous quadratic programming. *SIAM Journal on Discrete Mathematics* 12, 4, 500–523.

W. W. Hager, D. T. Phan, and H. Zhang. 2013. An exact algorithm for graph partitioning. *Mathematical Programming* 137, 1–2, 531–556.

M. T. Heath. 2002. *Scientific Computing: An Introductory Survey*. McGraw-Hill. http://books.google.com/books?id=DPkYAQAAIAAJ.

M. Holtgrewe, P. Sanders, and C. Schulz. 2010. Engineering a scalable high quality graph partitioner. In *Proceedings of the 24th IEEE International Symposium on Parallel and Distributed Processing.* 1–10.

Y. F. Hu and J. A. Scott. 2001. A multilevel algorithm for wavefront reduction. *SIAM Journal on Scientific Computing* 23, 4, 1352–1375. DOI:http://dx.doi.org/10.1137/S1064827500377733

S. E. Karisch, F. Rendl, and J. Clausen. 2000. Solving graph bisection problems with semidefinite programming. *INFORMS Journal on Computing* 12, 3, 177–191. DOI:http://dx.doi.org/10.1287/ijoc.12.3.177.12637

G. Karypis and V. Kumar. 1995. Analysis of multilevel graph partitioning. In *Proceedings of the 1995 ACM/IEEE Conference on Supercomputing (Supercomputing'95)*. Article No. 29.

J. Lescovec. n.d. Stanford Network Analysis Package (SNAP). Retrieved October 20, 2014, from http://snap.stanford.edu/index.html.

J. Maue and P. Sanders. 2007. Engineering algorithms for approximate weighted matching. In *Experimental Algorithms*. Lecture Notes in Computer Science, Vol. 4525. Springer, 242–255. http://dx.doi.org/10.1007/978-3-540-72845-0_19

H. Meyerhenke, B. Monien, and T. Sauerwald. 2008. A new diffusion-based multilevel algorithm for computing graph partitions of very high quality. In *Proceedings of the IEEE International Symposium on Parallel and Distributed Processing (IPDPS'08).* 1–13.

F. Pellegrini. Scotch Home Page. n.d. Retrieved October 20, 2014, from http://www.labri/fr/perso/pelegrin/scotch/.

A. Pothen, H. D. Simon, and K.-P. Liou. 1990. Partitioning sparse matrices with eigenvectors of graphs. *SIAM Journal on Matrix Analalysis and Applications* 11, 3, 430–452. DOI:http://dx.doi.org/10.1137/0611030

D. Ron, I. Safro, and A. Brandt. 2011. Relaxation-based coarsening and multiscale graph organization. *Multiscale Modeling & Simulation* 9, 1, 407–423.

D. Ron, S. Wishko-Stern, and A. Brandt. 2005. *An Algebraic Multigrid Based Algorithm for Bisectioning General Graphs*. Technical Report MCS05-01. Department of Computer Science and Applied Mathematics, Weizmann Institute of Science.

I. Safro, D. Ron, and A. Brandt. 2006. Graph minimum linear arrangement by multilevel weighted edge contractions. *Journal of Algorithms* 60, 1, 24–41.

I. Safro, D. Ron, and A. Brandt. 2008. Multilevel algorithms for linear ordering problems. *Journal of Experimental Algorithmics* 13, 1.4–1.20.

I. Safro, P. Sanders, and C. Schulz. n.d. Benchmark with Potentially Hard Graphs for Partitioning Problem. Retrieved October 20, 2014, from http://www.cs.clemson.edu/∼isafro/hardpart.html.

P. Sanders and C. Schulz. 2011. Engineering multilevel graph partitioning algorithms. In *Algorithms—ESA 2011*. Lecture Notes in Computer Science, Vol. 6942. Springer, 469–480.

P. Sanders and C. Schulz. 2012. Distributed evolutionary graph partitioning. In *Proceedings of the 12th Workshop on Algorithm Engineering and Experimentation (ALENEX'12)*. 16–29.

K. Schloegel, G. Karypis, V. Kumar, J. Dongarra, I. Foster, G. Fox, K. Kennedy, and A. White. 2000. *Graph Partitioning for High Performance Scientific Simulations*. Morgan Kaufmann.

E. Sharon, A. Brandt, and R. Basri. 2000. Fast multiscale image segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 70–77. citeseer.nj.nec.com/sharon99fast.html.

A. J. Soper, C. Walshaw, and M. Cross. 2004. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. *Journal of Global Optimization* 29, 2, 225–241.

L. E. Stock. 2006. *Strategic Logistics Management*. Lightning Source, La Vergne, TN. http://books.google.com/books?id=1LyCAQAACAAJ.

L. Tang, H. Liu, J. Zhang, and Z. Nazeri. 2008. Community evolution in dynamic multi-mode networks. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'08)*. 677–685.

U. Trottenberg and A. Schuller. 2001. *Multigrid*. Academic Press, Orlando, FL.

C. Walshaw. 2004. Multilevel refinement for combinatorial optimisation problems. *Annals of Operations Research* 131, 1, 325–372.

C. Walshaw and M. Cross. 2007. JOSTLE: Parallel multilevel graph-partitioning software—an overview. In *Mesh Partitioning Techniques and Domain Decomposition Techniques*, F. Magoules (Ed.). Civil-Comp Ltd., 27–58.

Received October 2012; revised May 2013; accepted September 2014