Combinatorial Preconditioners and Multilevel Solvers for Problems in Computer Vision and Image Processing*

Ioannis Koutis Gary L. Miller David Tolliver

Computer Science Department
Carnegie Mellon University

Abstract. Linear systems and eigen-calculations on symmetric diagonally dominant matrices (SDDs) occur ubiquitously in computer vision, computer graphics, and machine learning. In the past decade a multitude of specialized solvers have been developed to tackle restricted instances of SDD systems for a diverse collection of problems including segmentation, gradient inpainting and total variation. In this paper we explain and apply the support theory of graphs, a set of of techniques developed by the computer science theory community, to construct SDD solvers with provable properties. To demonstrate the power of these techniques, we describe an efficient multigrid-like solver which is based on support theory principles. The solver tackles problems in fairly general and arbitrarily weighted topologies not supported by prior solvers. It achieves state of the art empirical results while providing robust guarantees on the speed of convergence. The method is evaluated on a variety of vision applications.

1 Introduction

The Laplacian operator ∇^2 has played a central role in computer vision for nearly 40 years. In Horn's early work, he employed finite element methods for elliptical operators in shape from shading [20], to produce albedo maps [21], and flow estimates [22]. In Witkin's seminal work [39] he studied the diffusion properties of matrix equations derived from ∇^2 for linear filtering, later generalized by Perona and Malik [30] to the anisotropic case.

In recent years, combinatorial Laplacians of graphs have formed the algorithmic core of spectral methods [34, 29, 2, 40, 10, 15, 27, 11], random walks segmentation [15], in-painting [36, 28, 4], and matting methods [27]. Given the power of modern iterative solvers, we believe that reducing traditional image processing problems, such as Grady *et al*.'s work [17] on Mumford-Shah segmentation, to SDD systems at the inner loop is a critical endeavor. To this end, we note that non-linear filtering operations such as ℓ_2, ℓ_1 Total Variation [31, 9] and Non-Local Means [8, 7] can also be formulated as optimizations with these linear systems at their core. Further, we provide timing and modern complexity bounds for computer vision methods in §4 that require the solutions to SDD systems at their core.

* This work was partially supported by the National Science Foundation under grant number CCF-0635257 and the University of Pittsburgh Medical Center under award number A-006461.

From a practical standpoint, modern photos and videos, and medical images derived NMR and CT scanners provide enormously detailed portraits of a scene. As the resolution of imaging hardware has pushed at the limits of computational feasibility, researchers inevitably arrived at the study of iterative and **hybrid** solvers. Recently, vision and graphics researchers have developed specialized solvers [36, 13, 4, 28], and heuristic solvers with impressive empirical performance [33, 28, 18, 16]. In either case, the methods place strict requirements on the system, such as unit weight edges or 4-connectivity. For the methods that handle general weights, including Algebraic Multigrid (AMG) [32, 6], the solvers are based on heuristics and offer no guarantees on the speed of convergence. Indeed many applications, such as the spectral segmentation and convex programming, require wildly varying weights and often employ randomly sampled and loosely localized topologies. Furthermore, the heuristic nature of the solvers is generally undesirable in certain commercial applications, e.g. medical, where robust and timely behavior is a critical issue.

Given modern data volumes and reliability requirements it is clear that a SDD solver with provable convergence properties and sound theoretical machinery is important for the advancement and real-world success of methods based on linear system solutions. In this work we introduce the Combinatorial Multigrid Solver (CMG), a state of the art solver with provable properties. The CMG solver is based on principles of *support theory* for graphs, a set of techniques developed for the construction of *combinatorial preconditioners*, i.e. graphs that are simpler than a given graph and approximate it well in a precisely defined sense. An ancillary goal of this paper is to review certain useful fragments of support theory and apply them to analyze solvers.

2 Support Theory for graphs

Support Theory was developed for the study of Combinatorial subgraph preconditioners, introduced by Vaidya [38, 23]. It has been at the heart of impressive theoretical results which culminated in the work of Spielman and Teng [35] who demonstrated that SDD systems can be solved in nearly-optimal $\tilde{O}(n \log^{O(1)} n)$ time and later in the work of Koutis and Miller [25] who formally proved that SDD matrices with planar connection topologies (*e.g.* 4-connectivity in the image plane) can be solved asymptotically optimally, in $O(n)$ time. We dub these solvers **hybrid solvers** since they combine algorithms and ideas from direct solvers, preconditioned Conjugate Gradient, and recursion.

2.1 Reduction of SDDs to Laplacians

A matrix A is SDD if it is real symmetric and $A_{ii} \geq \sum_{j \neq i} |A_{ij}|$ for $1 \leq i \leq n$. The *Laplacian* A of a graph $G = (V, E, w)$, where w is a non-negative weight function on the edges, is defined by $A_{i,j} = A(j, i) = -w_{i,j}$ and $A_{i,i} = \sum_{i \neq j} w_{i,j}$. Thus Laplacians are SDD matrices having non-positive off diagonals and zero row sums. We briefly describe how any SDD system can be reduced to a Laplacian. SDD systems with positive off-diagonals can be reduced to the case of non-positive off diagonals using a very light-weight reduction known as the double-cover construction [19]. Assuming

now negative off-diagonals, nodes with positive row sums can be viewed as nodes that have an implicit edge to a new "grounded node". In general they do not cause any significant changes in the Laplacian solver [5, 24].

2.2 Preconditioners - Motivating Support Theory

Iterative algorithms, such as the Chebyshev iteration or the Conjugate Gradient, converge to a solution using only matrix-vector products with A. It is well known that iterative algorithms suffer from slow convergence properties when the conditioning of A, $\kappa(A)$, - defined as the ratio of the largest over the minimum eigenvalue of A - is large [1].

Preconditioned iterative methods attempt to remedy the problem by changing the linear system to $B^{-1}Ax = B^{-1}b$. In this case, the algorithms use matrix-vector products with A, and solve linear systems of the form $By = z$. The speed of convergence now depends on the **condition number** $\kappa(A, B)$, defined as

$$\kappa(A, B) = \max_x \frac{x^T Ax}{x^T Bx} \cdot \max_x \frac{x^T Bx}{x^T Ax} \tag{1}$$

where x is taken to be outside the null space of A. In constructing a preconditioner B, one has to deal with two contradictory goals: (i) Linear systems in B must be easier than those in A to solve, (ii) The condition number must be small to minimize the number of iterations.

Historically, preconditioners were natural parts of the matrix A. For example, if B is taken as the diagonal of A we get the Jacobi Iteration, and when B is the upper triangular part of A, we get the Gauss-Seidel iteration.

The cornerstone of combinatorial preconditioners is the following intuitive yet paradigm-shifting idea explicitly proposed by Vaidya: *A preconditioner for the Laplacian of a graph A should be the Laplacian of a simpler graph B, derived in a principled fashion from A.*

2.3 Graphs as electric networks - Support basics

There is a fairly well known analogy between graph Laplacians and resistive networks [12]. If G is seen as an electrical network with the resistance between nodes i and j being $1/w_{i,j}$, then in the equation $Av = i$, if v is the vector of voltages at the node, i is the vector of currents. Also, the quadratic form $v^T Av = \sum_{i,j} w_{i,j}(v_i - v_j)^2$ expresses the *power dissipation* on G, given the node voltages v. In view of this, the construction of a good preconditioner B amounts to the construction of a simpler resistive network (for example by deleting some resistances) with an energy profile close to that of A.

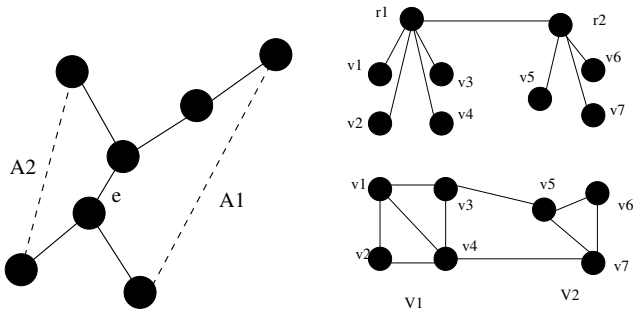
The **support** of A by B, defined as $\sigma(A/B) = \max_v v^T Av / v^T Bv$ is the number of copies of B that are needed to support the power dissipation in A, for all settings of voltages. The principal reason behind the introduction of the notion of support, is to express its local nature, captured by the Splitting Lemma.

Lemma 1 (Splitting Lemma). *If $A = \sum_{i=1}^m A_i$ and $B = \sum_{i=1}^m B_i$, where A_i, B_i are Laplacians, then $\sigma(A, B) \leq max_i \sigma(A_i, B_i)$.*

The Splitting Lemma allows us to bound the support of A by B, by splitting the power dissipation in A into small local pieces, and "supporting" them by also local pieces in B.

For example, in his work Vaidya proposed to take B as the maximal weight spanning tree of A. Then, it is easy to show that $\sigma(B, A) \leq 1$, intuitively because more resistances always dissipate more power. In order to bound $\sigma(A, B)$, the basic idea to let the A_i be edges on A (the ones not existing in B), and let B_i be the unique path in the tree that connects the two end-points of A_i. Then one can bound separately each $\sigma(A_i, B_i)$. In fact, it can be shown that any edge in A that doesn't exist in B, can be supported *only* by the path B_i.

As a toy example, consider the example in Figure 1(a) of the two (dashed) edges A_1, A_2 and their two paths in the spanning tree (solid) that share one edge e.



(a) A graph and its spanning tree - obtained by deleting the dashed edges.

(b) A graph and its Steiner preconditioner

Fig. 1.

In this example, the **dilation** of the mapping is equal to 3, i.e. the length of the longest of two paths. Also, as e is uses two times, we say that the **congestion** of the mapping is equal to 2. A core Lemma in Support Theory [3, 5] is that the support can be upper bounded by the product **congestion**$*$**dilation**.

2.4 Steiner preconditioners

Steiner preconditioners, introduced in [19] and extended in [26] introduce external nodes into preconditioners. The proposed preconditioner is based on a partitioning of the n vertices in V into m vertex-disjoint clusters V_i. For each V_i, the preconditioner contains a star graph S_i with leaves corresponding to the vertices in V_i rooted at a vertex r_i. The roots r_i are connected and form the **quotient** graph Q. This general setting is illustrated in Figure 1(b), consisting of good clusters.

Let D' be the total degree of the leaves in the Steiner preconditioner S. Let the **restriction** R be an $n \times m$ matrix, where $R(i, j) = 1$ if vertex i is in cluster j and 0

otherwise. Then, the Laplacian of S has $n + m$ vertices, and the algebraic form

$$S = \begin{pmatrix} D' & -D'R \\ -R^T D' & Q + R^T D' R \end{pmatrix}. \tag{2}$$

A worrying feature of the Steiner preconditioner S is the extra number of vertices. So how do we even use it? Gremban and Miller [19] proposed that every time a system of the form $Bz = y$ is solved in an usual preconditioned method, the system $S \begin{pmatrix} z \\ z' \end{pmatrix} = \begin{pmatrix} y \\ 0 \end{pmatrix}$ should be solved instead, for a set of *don't care* variables z'. They also showed that the operation is equivalent to preconditioning with the dense matrix

$$B = D' - V(Q + D_Q)^{-1} V^T \tag{3}$$

where $V = D'R$, and $D_Q = R^T D' R$. The matrix B is called the Schur complement of S with respect to the elimination of the roots r_i, further it is a well known that B is also a Laplacian.

The analysis of the support $\sigma(A/S)$, is identical to that for the case of subgraph preconditioners. For example, going back to Figure 1(b), the edge (v_1, v_4) can only be supported by the path (v_1, r_1, v_4), and the edge (v_4, v_7) only by the path (v_4, r_1, r_2, v_7). Similarly we can see the mappings from edges in A to paths in S for every edge in A. In the example, the **dilation** of the mapping is 3, and it can be seen that to minimize the **congestion** on every edge of S (i.e. make it equal to 1), we need to take $D' = D$, where D are the total degrees of the nodes in A, and $w(r_1, r_2) = w(v_3, v_5) + w(v_4, v_7)$. More generally, for two roots r_i, r_j we should have $w(r_i, r_j) = \sum_{i' \in V_i, j' \in V_j} w_{i,j}$. Under this construction, the algebraic form of the quotient Q can be seen to be $Q = R^T A R$.

So far no special properties of the clustering have been used. Those come into play in bounding the support of S by A, $\sigma(S/A)$. In [26] it was shown that the support $\sigma(S/A)$ reduces to bounding the support $\sigma(S_i, A[V_i])$, for all i, where $A[V_i]$ denotes the graph induced in A by the vertices V_i. When are these bounded? Before we answer this question, let us recall the definition of **conductance**.

Definition 1. *The conductance $\phi(A)$ of a graph $A = (V, E, w)$ is defined as*

$$\phi(A) = \min_{S \subseteq V} \frac{w(S, V - S)}{\min(w(S), w(V - S))}$$

where $w(S, V - S)$ denotes the total weight connecting the sets S and $V - S$, and where $w(S)$ denotes the total weight incident to the vertices in S.

The main result of [26] is captured by the following Theorem.

Theorem 1. *The support $\sigma(S/A)$ is bounded by a constant c independent from n, if and only for all i the conductance of the graph $A^o[V_i]$ induced by the nodes in V_i augmented by the edges leaving V_i is bounded by a constant c'.*

Although Theorem 1 doesn't give a way to pick to clusters, it does provide a way to *avoid* bad clusterings.

2.5 Support Theory & Grady's clusterings

In recent work [16], Grady proposed a multigrid method where the construction of the "coarse" grid follows exactly the construction of the **quotient** graph in the previous section. Specifically, Grady proposes a clustering such that every cluster contains exactly one of certain pre-specified "coarse" nodes. He then defines the restriction matrix R and he lets the coarse grid be $Q = R^T AR$, identically to the construction of the previous Section. The question then is whether the proposed clustering provides the guarantees that by Theorem 1 are necessary to construct a good Steiner preconditioner. In the following Figure, we replicate Figure 2 of [16], with a choice of weights that force the depicted clustering.

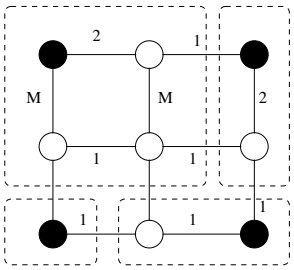


Fig. 2. A bad clustering

Every cluster in Figure 2 contains exactly one black/coarse node. The problem with the clustering is that the top left cluster, has a very low conductance when $M >> 1$. In general, in order to satisfy the requirement of Theorem 1, there are cases where the clustering has to contain clusters with *no* coarse nodes in them.

It is interesting that the Maximally Connected Neighbor (MCN) algorithm proposed in [16] comes very close to the clustering algorithm proposed in [26]. Of course, it is imaginable that there are instances where MCN may not induce bad clusterings. On such instances, Grady's clustering has provable properties. Grady's solver is a multigrid solver, but as we will see in Section 3, multigrid solvers and Steiner preconditioners are closely related.

3 The Combinatorial Multigrid Solver

In this section we describe the Combinatorial Multigrid Solver. As we will see, the CMG solver matches the simple form of AMG, but with two distinguishing features: (i) The "coarsening" strategy is markedly different; it is in fact easier to implement and faster than the various AMG coarsening strategies. (ii) The algorithm is truly "black-box", in stark contrast which AMG which employs an extensive list of algorithmic knobs.

3.1 A graph decomposition algorithm

According to the discussion of §2.4, the crucial step for the construction of a good Steiner preconditioner is the computation of a group decomposition that satisfies, as best as possible, the requirements of Theorem 1. Before the presentation of the **Decompose-Graph** algorithm, that extends the ideas of [26], we need to introduce a couple of definitions. Let $vol_G(v)$ denote the total weight incident to node v in graph G. The *weighted degree* of a vertex v is defined as the ratio

$$wd(v) = \frac{vol(v)}{\max_{u \in N(v)} w(u,v)}.$$

The *average weighted degree* of the graph is defined as $awd(G) = (1/n) \sum_{v \in V} wd(v)$.

Algorithm **Decompose-Graph**

Input: Graph $A = (V, E, w)$
Output: Disjoint Clusters V_i with $V = \bigcup_i V_i$

1. Let $W \subseteq V$ be the set of nodes satisfying $wd(v) > \kappa \cdot awd(A)$, for some constant $\kappa > 4$.
2. Form a forest graph F, by keeping the heaviest incident edge of v for each vertex $v \in V$ in A.
3. For every vertex $w \in W$ such that $vol_T(w) < vol_G(w)/awd(A)$ remove from F the edge contributed by w in Step 2.
4. Decompose each tree T in F into vertex-disjoint trees of constant conductance.

It is not very difficult to prove that the algorithm **Decompose-Graph** produces a partitioning where the conductance of each cluster depends only on $awd(A)$ and the constant κ. In fairly general topologies that allow high degree nodes, $awd(A)$ is constant and the number of clusters m returned by the algorithm is such that $n/m > 2$ (and in practice larger than 3 or 4). There are many easy ways to implement Step 3. Our current implementation makes about three passes of A. Of course, one can imagine variations of the algorithm (i.e. a correction step, etc) that may make the clustering phase a little more expensive with the goal of getting a better conductance and an improved condition number, if the application at hand requires many iterations of the solver.

3.2 From Steiner preconditioners to Multigrid

Multigrid algorithms have been a very active research area for nearly three decades. There are many expository article and books, among which [37]. In order to describe the reasoning that leads to our Combinatorial Multigrid Algorithm, we will need to shortly review the basic principles behind the generic two-level iteration.

Algebraically, any of the classic preconditioned iterative methods, such as the Jacobi and Gauss-Seidel iteration, is nothing but a matrix $\mathcal{S}$, which gets applied implicitly to the current error vector e, to produce a new error vector $e' = \mathcal{S}e$. For example, in the

Jacobi iteration we have $\mathcal{S} = (I - D^{-1}A)$. This has the effect that it reduces effectively only part of the error in a given iterate, namely the components that lie in the low eigenspaces of $\mathcal{S}$ (usually referred to as high frequencies of A). The main idea behind a two-level multigrid is that the current *smooth* residual error $r = b - Ax$, can be used to calculate a correction $P^T Q^{-1} P r$, where Q is a smaller graph and P is an $m \times n$ restriction operator. The correction is then added to the iterate x. The hope here is that for smooth residuals, the low-rank matrix $P^T Q^{-1} P$ is a good approximation of A^{-1}. Algebraically, this correction is the application of the operator $T = (I - P^T Q^{-1} P A)$ to the error vector e. The choice of P and Q is such that T is a projection operator with respect to the A-inner product, a construction known as the *Galerkin* condition. Two-level convergece proofs are then based on bounds on the angle between the subspace $Null(P)$ and the high frequency subspace of $\mathcal{S}$.

At a high level, the key idea behind CMG is that the provably small condition number $\kappa(A, B)$ where B is given in expression 3, is equal to the condition number $\kappa(\hat{A}, \hat{B})$ where $\hat{A} = D^{-1/2} A D^{-1/2}$ and $\hat{B} = D^{-1/2} B D^{-1/2}$. This in turn implies a bound on the angle between the low frequency of $\hat{A}$ and the high frequency of $\hat{B}$ [26]. The latter subspace includes $Null(R^T D^{1/2})$. This fact suggests to choose $R^T D^{1/2}$ as the projection operator while performing relaxation with $(I - \hat{A})$ on the system $\hat{A} y = D^{-1/2} b$, with $y = D^{1/2} x$. Combining everything, we get the following two-level algorithm.

Two-level Combinatorial Multigrid

Input: Laplacian $A = (V, E, w)$, vector b, approximate solution x, $n \times m$ restriction matrix R
Output: Updated solution x for $Ax = b$

1. $D := diag(A)$; $\hat{A} := D^{-1/2} A D^{-1/2}$;
2. $z := (I - \hat{A}) D^{1/2} x + D^{-1/2} b$;
3. $r := D^{-1/2} b - \hat{A} z$; $w := R^T D^{1/2} r$;
4. $Q := R^T A R$; Solve $Qy = w$;
5. $z := z + D^{1/2} R y$
5. $x := D^{-1/2}((I - \hat{A}) z + D^{-1/2} b) z$

The two-level algorithm can naturally be extended into a full multigrid algorithm, by recursively calling the algorithm when the solution to the system with Q is requested.

4 Experiments

Many computer vision problems naturally suggest a graph structure - for example the vertices often correspond to samples (*e.g.* pixels, patches, images), the edge set establishes pairwise comparisons or constraints encoded in the graph and the weights are either data driven (for clustering) or the result of an ongoing optimization procedure (weights in the t^{th} iteration of Newton's method).

In this section we demonstrate our general case combinatorial multigrid preconditioner (CMG) solver on a suite of applications taken from computer vision. The data is presented relative to the timing of a direct methods, found in MATLAB and LAPACK, for reference. We emphasize that our goal is not to perform numerical comparisons

with any of the previous solvers, but rather to demonstrate that fast solvers with provable properties (such as CMG) are within the realm of practical implementation. The presented solver was written in a combination of `C` and `MATLAB` with modest attention paid to code optimization.

Notation The weighted laplacian matrix A can be factored as $A = \Gamma^T W \Gamma$ where Γ is a edge-node incidence matrix, and W is a nonnegative diagonal weight matrix over the edges. The normalized Laplacian is defined $\hat{A} = D^{-1/2} A D^{-1/2} = I - D^{-1/2} G D^{-1/2}$ where D is the weighted degree matrix and G the weighted adjacency matrix. Recall that quadratic form $x^T Ax$ can be written in several forms $x^T Dx - x^T Gx = \sum_i d_i x_i^2 - 2\sum_{ij} w_{ij} x_i x_j = \sum_{ij} w_{ij}(x_i - x_j)^2$.

4.1 SDD Linear Systems

Sparse, unit weight, SDD linear systems arise in non-local means[8, 7], gradient inpainting [36, 28, 4], segmentation [14], regression and classification [41] and related data interpolation optimizations. For example, the in-painting functional $f(x) = min_x : (\Gamma x - \Delta)^T W (\Gamma x - \Delta)$, where Δ is a vector of target gradient values to be exhibited in the image x and Γ is a generalized gradient operator, requires the solution to $\Gamma^T W \Gamma x = \Gamma^T W \Delta$. When W is the identity and Γ embodies a 4-connected topology these systems can be efficiently solved (provably) by geometric multigrid methods including the method described herein. For unit and weighted general planar systems the asymptotic complexity of solving $Ax = b$ is $O(n)$ [26].

When W is a more general nonnegative diagonal matrix and Γ encodes non-planar connectivity many multigrid methods will fail. Figure 3 shows such a class of problems in a log-log plot for weighted 3D graphs derived from a 3D CT Study of an oncology phantom – with the edge weighting: $w_{uv} = exp\left(-(I_u - I_v)^2/\sigma^2\right)$ between neighboring voxels u and v, where I_u denotes the intensity of voxel u. Solving such 3D lattice SDDs, and general topologies, requires only $O(nlog^{o(1)}n)$ [35] work, however we observe linear work empirically.

Eigencalculations: Calculating a minimal, say $k-$dimensional, eigenspace of an SDD matrix forms the computational core of the spectral relaxation for NCuts [34], spectral clustering [29], Laplacian eigenmaps [2], diffusion maps [10], and the typical case for Levin *et. al.*'s image matting algorithm [27]. Recall that eigensystems satisfy the following equations for a Laplacian $Az_i = \phi_i z_i$ and generalize to $Ax_i = \lambda_i D x_i$ where we assume D is positive definite diagonal in general, and typically D is the weighted degree matrix (see §2.4).

For the generalized problem, efficient computation of the k eigenpairs (x_i, λ_i), such that $\lambda_1 \leq \lambda_i \leq \lambda_{k+1}$, depends upon the relationship to a normalized Laplacian, $\hat{A}$. $\hat{A}$ is possessed of the same eigenvalues $\{\lambda_i\}$ as the generalized problem, with eigenvectors y_i that map to generalized vectors under the operator $D^{1/2}$: $x_i = D^{1/2} y_i$.

We find the set of eigenvalues and vectors by inverse powering, *i.e.* repeated solves of the problem $\hat{A} q^{t+1} = q^t$, coupled with a Krylov space method such as the Lanczos algorithm. In [35] the number of inverse powers required was shown to be $O(\log n)$

to calculate a vector with a Rayleigh quotient arbitrarily close to that of the minimal eigenspace. Thus the general case complexity of an eigencalculation remains $O(n \log^{o(1)} n)$. In essence by employing the CMG solver we achieve approximate NCuts solutions in time roughly proportional to sorting the vertices (pixels) by intensity. Timing results for estimating (x_2, λ_2) on three panoramic landscape image derived graphs is shown in column 4 of Table 1 (edge weightings as in above).

Convex Programming and Reweighted Problems: The optimization of convex functionals such as "$\ell_2 - \ell_1$ Total Variation"[31], given by: $f(x) = min_x \ : \ ||x - s||_2 + \lambda|\nabla x|_1$ for an input signal s, and related problems can be accomplished using Newton's method with log-barriers in $O(n^{1.5}\ log^{o(1)}(n))$ time with modern solvers. For TV, the computational crux of each iteration is the solution of a SDD system with iteration dependent edge weights for the Laplacian modeling the ℓ_1 penalty on spatial gradients $|\nabla x|_1$. As the program iterates, the weights on edges between regions of different intensity approach zero; such weightings radically violate the conditions required by most multi-grid methods. CMG is suitable for such weightings, and further, the clustering determined by the steps in §3 are generally preserved across iterations thus requiring linear work to update the preconditioner, unlike, direct methods which require recomputing the factorization from scratch. This property is also useful for other iterative methods such as robust least squares. Timing results on the panoramic image problems can be found in columns 5 and 6 of Table 1.

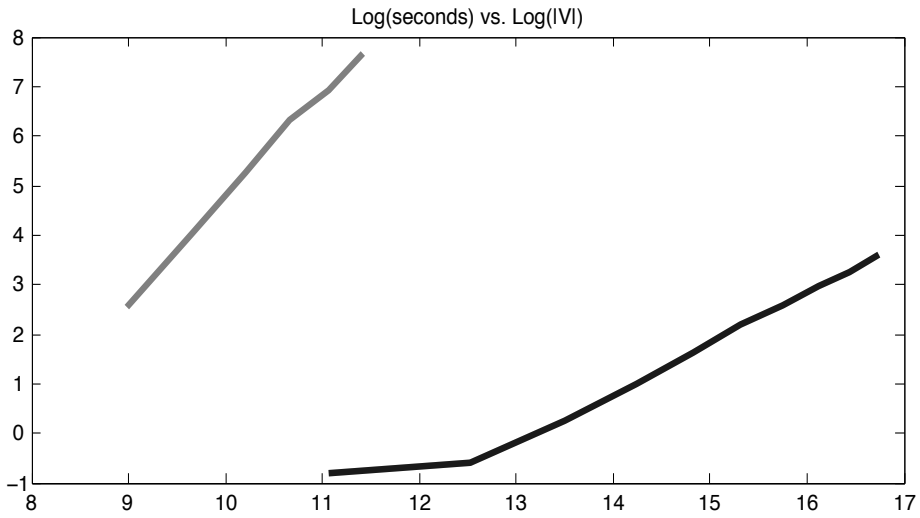


Fig. 3. Relative speeds of the CMG solver (black) and MATLAB's "\" operator (grey) on weighted graphs with 6 connected 3D lattices derived from a CT study. The X-axis is shown in log scale over $|V|$, the number of variables in the system. For reference, the execution time for the CMG solver on a problem with 27 million variables is $\approx$ 50 seconds.

Size in Mega-pixels	"\"	CMG	"\"+*eigs*	CMG+*eigs*	ℓ_2, ℓ_1 w "\"	ℓ_2, ℓ_1 w CMG
2M	45s	**8s**	1.6m	**59s**	5.1m	**31s**
10M	4.9m	**22s**	7.6m	**2.7m**	49.4m	**3.7m**
50M	*NA*	**1.1m**	*NA*	**7.1m**	*NA*	**14.6m**

Table 1. Two dimensional comparison with MATLAB for solutions on weighted 2D problems (with $O(|V|)$ super-lattice topology) at 2, 10, and 50 megapixels. The construction of the CMG hierarchy takes less than 5 seconds for the $50M$ problem. For CMG, the majority of the compute time is consumed by the solver iteration, for direct methods computing the factorization is most time consuming.

5 Discussion

Finally, by segregating the code for solvers (and eigen-calculations) from that of the applications we harvest improved modularity, reliability and factorizations of the system. Thus, 1) errors can be isolated to either the solver or the application, 2) as new solvers become available they can be easily adopted and perhaps most importantly 3) it relieves the application designer of the burden of implementing the state-of-the-art in solver technology.

We feel that significant improvements are still to be made in the solver and eigen-solver technology. We envision major new applications for these solvers in scientific computing, image processing, data-mining, and machine learning. Finally, to ease adoption of **hybrid** solvers, an implementation of the CMG solver will be made available in the near future.

References

1. O. Axelsson. *Iterative Solution Methods*. Cambridge University Press, New York, NY, 1994.
2. M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396.
3. M. Bern, J. R. Gilbert, B. Hendrickson, N. Nguyen, and S. Toledo. Support-graph preconditioners. *SIAM J. Matrix Anal. Appl.*, 4:930–951, 2006.
4. P. Bhat, B. Curless, M. Cohen, and C. L. Zitnick. Fourier analysis of the 2d screened poisson equation for gradient domain problems. In *ECCV*, 2008.
5. E. G. Boman and B. Hendrickson. Support theory for preconditioning. *SIAM J. Matrix Anal. Appl.*, 25(3):694–717, 2003.
6. A. Brandt. General highly accurate algebraic coarsening. 2000.
7. T. Brox, O. Kleinschmidt, and D. Cremers. Efficient nonlocal means for denoising of textural patterns. *Trans. on Image Processing*.
8. A. Buades, B. Coll, and J. Morel. Nonlocal image and movie denoising. *IJCV*, 76(2):123–139, 2008.
9. T. Chan and J. Shen. *Image Processing And Analysis: Variational, Pde, Wavelet, And Stochastic Methods*. SIAM, 2005.
10. R. R. Coifman, S. Lafon, A. B. Lee, M. Maggioni, B. Nadler, F. Warner, and S. W. Zucker. Diffusion maps geometric diffusions as a tool for harmonic analysis and structure definition of data. *PNAS*, 102(21):7426–7431, May 2005.
11. T. Cour and J. Shi. Solving markov random fields with spectral relaxation. *AISTATS*, 2007.
12. P. G. Doyle and J. L. Snell. Random walks and electric networks, 2000.
13. G. Gilboa and S. Osher. Nonlocal operators with applications to image processing. July 2007.

14. L. Grady. Multilabel random walker image segmentation using prior models. In *CVPR*, volume 1 of *CVPR*, pages 763–770, San Diego, June 2005. IEEE, IEEE.
15. L. Grady. Random walks for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 2(11):1768–1783, 2006.
16. L. Grady. A lattice-preserving multigrid method for solving the inhomogeneous poisson equations used in image analysis. *ECCV*, 5303:252–264, 2008.
17. L. Grady and C. Alvino. Reformulating and optimizing the mumford-shah functional on a graph - a faster, lower energy solution. *ECCV*, 5302:248–261, 2008.
18. L. Grady and A. K. Sinop. Fast approximate random walker segmentation using eigenvector precomputation. In *CVPR*, 2008.
19. K. Gremban. *Combinatorial Preconditioners for Sparse, Symmetric, Diagonally Dominant Linear Systems*. PhD thesis, Carnegie Mellon University, Pittsburgh, October 1996. CMU CS Tech Report CMU-CS-96-123.
20. B. Horn. Shape from shading: A method for obtaining the shape of a smooth opaque object from one view. Technical Report 232, MIT AI Laboratory, November 1970.
21. B. Horn. Determining lightness from an image. *Computer Graphics and Image Processing*, 3(1):277–299, 1974.
22. B. Horn. Determining optical flow. *MIT AI Laboratory*, 17(1):185–203, 1981.
23. A. Joshi. *Topics in Optimization and Sparse Linear Systems*. PhD thesis, University of Illinois at Urbana Champaing, 1997.
24. I. Koutis. *Combinatorial and algebraic algorithms for optimal multilevel algorithms*. PhD thesis, Carnegie Mellon University, Pittsburgh, May 2007. CMU CS Tech Report CMU-CS-07-131.
25. I. Koutis and G. L. Miller. A linear work, $O(n^{1/6})$ time, parallel algorithm for solving planar Laplacians. In *SODA 2007*.
26. I. Koutis and G. L. Miller. Graph partitioning into isolated, high conductance clusters: Theory, computation and applications to preconditioning. In *SPAA*, 2008.
27. A. Levin, A. Rav-Acha, and D. Lischinski. Spectral matting. In *CVPR*, 2007.
28. J. McCann and N. S. Pollard. Real-time gradient-domain painting. *SIGGRAPH*, 27(3), 2008.
29. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *NIPS*, 2002.
30. P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *PAMI*, 7(12):629–639, 1990.
31. L. Rudin, S. Osher, and E. Fatemi. Nonlinear total variation based noise removal algorithm. *Physica D*, (60):259–268, 1992.
32. J. W. Ruge and K. Stüben. Algebraic multigrid (AMG). In S. F. McCormick, editor, *Multigrid Methods*, volume 3 of *Frontiers in Applied Mathematics*, pages 73–130. SIAM, Philadelphia, PA, 1987.
33. E. Sharon, A. Brandt, and R. Basri. Segmentation and boundary detection using multiscale intensity measurements. In *CVPR*, 2001.
34. J. Shi and J. Malik. Normalized cuts and image segmentation. In *PAMI*, 2000.
35. D. A. Spielman and S.-H. Teng. Nearly-linear time algorithms for preconditioning and solving symmetric, diagonally dominant linear systems, 2006.
36. R. Szeliski. Locally adapted hierarchical basis preconditioning. *SIGGRAPH*, 25(3):1135–1143, August 2006.
37. U. Trottenberg, A. Schuller, and C. Oosterlee. *Multigrid*. Academic Press, 1st edition, 2000.
38. P. M. Vaidya. Solving linear equations with symmetric diagonally dominant matrices by constructing good preconditioners. A talk based on this manuscript, October 1991.
39. A. Witkin. Scale-space filtering. *IJCAI*, pages 1019–1022, August 1983.
40. S. Yu and J. Shi. Segmentation given partial grouping constraints. *PAMI*, 26(2):173–183, 2004.
41. X. Zhu, Z. Ghahramani, and J. D. Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. *ICML*, 2003.