Università di Pisa
Dipartimento di Informatica

Technical Report: TR-10-02

Multi-iterative techniques of Multigrid type for solving large linear systems with structure of graph

Pietro Dell'Acqua Antonio Frangioni Stefano Serra Capizzano

January 13, 2010

address: Largo B. Pontecorvo 3, 56127 Pisa, Italy. tel: +39 050 2212700 fax: +39 050 2212726

Multi-iterative techniques of Multigrid type for solving large linear systems with structure of graph

Pietro Dell'Acqua * Antonio Frangioni † Stefano Serra Capizzano ‡

January 13, 2010

Abstract

We consider multi-iterative techniques of multigrid type for the numerical solution of large linear systems with (weighted) structure of graph. We combine proper coarser-grid operators with auxiliary techniques. We show that the most effective smoothers have to be of Krylov type with spanning tree preconditioners, while the projectors have to be designed for maintaining as much as possible the structure of graph matrix at the inner levels. Some necessary and sufficient conditions are proved; in this framework it is possible to explain why the classical projectors inherited from differential equations are good in the differential context and why they behave unsatisfactorily for unstructured graphs. Several numerical experiments have been conducted showing that our approach is effective even in very difficult cases where the known approaches are rather slow.

Keywords: *Graph Matrices, Multigrid, Conditioning and Preconditioning*

1 Introduction

The paper is concerned with solution methods for (extremely) large linear systems whose matrices have a graph structure with weighing on the arcs (see e.g. [13] and references therein). Such a type of matrices are encountered in several applications where a network structure is present, ranging from Web searching engines [25] to general Markov chains [38], from consensus algorithms [30] to optimization problems such as the Minimum Cost Flow (MCF) in networks [1, 5]. In all these cases the dimension of the involved matrices is very large to extremely large.

A typical example is given by the classic Google matrix [25]: indeed the Web can be regarded as a huge directed graph whose n nodes are all the Web pages and whose arcs are constituted by all the direct links between pages. Now the problem to measure the *importance* $y(j)$ of any given web page j and the idea is to measure it according to the following list of axioms:

- the page j is more important if there exists a page $i \neq j$ referring to it;
- the page j is more important if the importance $y(i)$, with $i \neq j$ referring j, is higher;
- the page j is less important if $i \neq j$, referring to j, links also to many other pages $k \neq i, j$.

*Dipartimento di Fisica e Matematica, Università degli Studi dell'Insubria, Via Valleggio 11, 22100 Como, Italy. E-mail: peterwater@interfree.it

†Dipartimento di Informatica, Università di Pisa, Polo Universitario della Spezia, Via dei Colli 90, 19121 La Spezia, Italy. E-mail: frangio@di.unipi.it

‡Dipartimento di Fisica e Matematica, Università degli Studi dell'Insubria, Via Valleggio 11, 22100 Como, Italy. E-mail: stefano.serrac@uninsubria.it

Remarkably, these axioms recall Andy Warhol's measure of social importance "Don't pay attention to what they write about you. Just measure it in inches" and its several distinguished precursors such as George Cohan's "I don't care what you say about me, as long as you say something about me, and as long as you spell my name right" and Oscar Wilde's "The only thing worse than being talked about is not being talked about". If $\deg(i)$ indicates the cardinality of the pages different from i which are reached by a direct link from page i, the importance $y(j)$ of any page $j = 1, \ldots, n$ could then be defined as

$$y(j) = \sum_{i \to j} \frac{y(i)}{\deg^*(i)} \quad , \quad y(j) \geq 0 \quad , \quad \sum_{i=1}^{n} y(i) = 1 \ ,$$

where $\deg^*(i) = \deg(i)$ if $\deg(i) > 0$ and $\deg^*(i) = n$ if $\deg(i) = 0$ to account for the so-called dangling nodes (the last constraint simply provides a reasonable normalization). The definition is nice in principle and it can be interpreted in matrix-vector terms as $y^T G = y^T$, that is, an eigenvalue/eigenvector problem on the simplest Google matrix G defined as $G_{ij} = 1/\deg^*(i)$. The problem can then be enriched with the use of a damping parameter [25, 22] and can be transformed into the solution of a linear system whose size is the number of web pages, that is of order of 10^{10} [14].

By this concise description it is evident that the Google problem can be considered a specific instance of Markov chains problems [38]. A somewhat different application, which is central to the development of this article at least as far as the computational part is concerned, are linear systems arising as intermediate steps in Interior Point (IP) techniques for MCF problems [1, 5]. In this setting, a linear system involving the weighted *Laplacian* of the underlying network has to be solved at each iteration, with varying vector of arc weights Θ (cf. (2)). Not only the networks can be very large, e.g. with up to 2^{22} arcs [31]; these approaches allow a more or less direct extension [9, 10] to *multicommodity* flow problems [16], that have a huge range of practical applications from telecommunication [12] to transportation [17] and beyond. In this case, the size of the matrix is further multiplied by the number of *commodities* (different types of flows in the network), that can be easily run into the thousands (e.g. being *quadratic* in the number of nodes in some applications); furthermore, the graph structure is somewhat "muddled" by the rows corresponding to mutual capacity constraints.

All the above mentioned classes of problems exhibit three main issues:

- the large size of the considered linear systems;
- the sparsity and the graph structure of the involved matrices;
- the potential ill-conditioning, as a function of the matrix dimension and/or of other critical parameters (such as the weights vector Θ).

These should immediately refrain from recommending direct solvers. For instance, the methods based on factorizations [21], such as Gaussian LR, QR or real Cholesky LL^T (only for the MCF problem), do not exploit the structure and in particular the sparsity of the matrices, so that the cost will become unacceptably high both in space and time [9] and the precision unacceptably poor in case of ill-conditioning.

The use of iterative solvers has the immediate advantage that the matrix vector product can be done linearly in case of sparsity; moreover no storage problems arise since the original coefficient matrix is not manipulated. However, the advantages in the reduced cost per iteration could be negated by a huge number of iterations (i.e., slow convergence, stagnation, non convergence). Finding fast convergent methods is non-trivial; ideally, one would like to identify *optimal* methods in which the number of iterations for reaching a pre-assigned accuracy $\varepsilon > 0$ is independent of the dimension and (possibly) of the other parameters involved in the problem, and only depends on the parameter ε (for a precise notion of optimality see [4]). This difficulty is typically related to ill-conditioning, i.e., the possible non normality of the matrix or ill-conditioning of the eigenvector matrix. In fact, if the matrix is well-conditioned, then simple scaled Richardson methods or scaled Richardson methods for normal equations are optimal. The same can be claimed for Krylov methods such as conjugate gradient (CG) or GMRES. However, for a matrix-sequence A_n whose condition number $g(n)$ diverges as $n \to \infty$, the classical iterative solvers (relaxed Richardson, relaxed Jacobi, relaxed Gauss-Seidel, CG etc.) are not convergent or, at best, the iteration count, as a function of n, diverges as well in a way that can be related with $g(n)$ [21, 33]. As an example, we consider the discrete Laplacian Δ_n

related to the equi-spaced finite difference approximation of $-d^2/dx^2$ on $(0,1)$ with homogeneous Dirichlet boundary conditions, which can also be viewed as a MCF matrix $E\Theta E^T$ where Θ is the identity and E is the node-arc incidence matrix of a linear graph, and the discrete bi-Laplacian Δ'_n related to the same type of approximation of the operator d^4/dx^4 with homogeneous boundary conditions [23]. Since $g(n) \sim n^2$ for Δ_n and $g(n) \sim n^4$ for Δ'_n, it is clear that the speed of convergence may well not be satisfactory; the following table reports the iteration count for different classical methods and for reaching a given accuracy.

	Jacobi/Richardson	Gauss-Seidel	CG
Δ_n	$O(n^2)$	$O(n^2)$	$O(n)$
Δ'_n	$O(n^4)$	$O(n^4)$	$O(n)$

A more complete analysis of the conditioning and extremal eigenvalues of more general graph matrices can be found in [20].

Three classes of approaches are known that could be successful for ill-conditioned systems:

- Krylov methods with preconditioning [33];
- multigrid methods (or multi-level methods, aggregation-disaggregation, etc.) [40];
- multi-iterative solvers [36].

In the case of graph matrices, the first approach has been considered mainly through *support-graph preconditioners*. Proposed by P. Vaidya in 1991 but not analyzed or implemented at that time, these are based on the idea of extracting a properly structured subgraph (such as a tree or tree-like structure) and using the corresponding matrix as preconditioner. Several authors have analyzed these preconditioners both from the theoretical and from the practical viewpoint (cf. [7, 8, 27, 18, 19] and the references therein).

Little is known, instead, about the application of specialized multigrid techniques for the problem, especially in the framework of multi-iterative solvers. The idea [36] is to define a method which can be viewed as the combination of different iterations, where the key notion is the spectral complementarity. In other words, each method could be slowly convergent in itself, but their combination is fast since each of them substantially reduces the error in a given subspace and the global direct sum of such subspaces coincides with $\mathbb{R}^n$ (or $\mathbb{C}^n$). Of course, the most difficult problem is to find an iteration which is rapidly convergent in the subspace where the original problem is ill-conditioned (that is related, after scaling, to the eigen-spaces associated with small eigenvalues). This task is achieved by projecting the problem in the ill-conditioned subspace via the coarse-grid correction and so reducing the problem-size. Furthermore, in order to have a recursive procedure, it is necessary to maintain the structure at the lower levels. While this problem has been satisfactorily solved in the context of differential equations and in the context of structured matrices related to fast transforms (see e.g. [40, 2] and references therein), there is no analysis in the case of general graph matrices. The main contribution of this paper is precisely the definition of a proper way for projecting a graph matrix into a graph matrix and to exploit the multi-iterative idea in order to get fast convergent methods. The goal is to treat efficiently the case of extremely ill-conditioned linear systems, where the preconditioned Krylov methods used so far are not sufficiently effective.

The paper is organized as follows. In Section 2 we introduce in more detail the MCF problem and we describe in general terms the multigrid procedure. Sections 3, 4 and 5 are devoted to the study of the right projectors in order to keep as much as possible the graph structure. Finally, Section 6 and Section 7 are devoted to a critical discussion of the numerical experiments and to final conclusions, respectively.

Notation 1 *For reader convenience we report a table with the acronyms used in the paper.*

MCF	Minimum Cost Flow
IP	Interior Point algorithm
CG	Conjugate Gradient
PCG	Preconditioned Conjugate Gradient
MGM	Multi Grid Method
AMG	Algebraic Multi Grid
FWO	Full Weighting Operator
AGO	Aggregation Operator
`Net`, `Grid`, `Goto`	Problem generators
$\mathbf{PCG_n}$	A particular PCG method
$\mathbf{MGM_n}$	A particular MGM method
$-$	Number of iterations higher than 100
$*$	Number of iterations higher than 5000

2 Preliminary steps

In this section we first present in some detail the MCF problem, which will serve as our primary benchmark for numerical tests, then we introduce the multigrid procedure to be specialized in the next sections in our graph setting.

2.1 The Minimum Cost Flow

We start by recalling the definition of *node-arc* incidence matrix of a directed graph.

Definition 1 *Let $\mathcal{G} \equiv \mathcal{G}_n = (\mathcal{U}_n, \mathcal{V}_n)$ be a* directed graph *with n nodes $\mathcal{U}_n = \{u_1, \ldots, u_n\}$ and m arcs $\mathcal{V}_n = \{v_1, \ldots, v_m\}$; its* node-arc incidence matrix *$E \equiv E_n = E(\mathcal{G}_n)$ is the $n \times m$ matrix such that $E_{ij} = 1$ if v_j emanates from u_i, $E_{ij} = -1$ if v_j terminates at u_i and $E_{ij} = 0$ otherwise. Hence, E has exactly two non-zero elements (a 1 and a -1) in every column.*

Given a directed graph $\mathcal{G}$, the linear Minimum Cost Flow (MCF) problem [5] is the Linear Program (LP)

$$\min \left\{ c^T x \ : \ Ex = d \ , \ 0 \le x \le u \right\} \tag{1}$$

where E is the node-arc incidence matrix of $\mathcal{G}$, c is the vector of arc costs, u is the vector of arc upper capacities, d is the vector of node deficits and x is the vector of flows. The *flow conservation constraints* $Ex = d$ express the fact that the flow has to travel in the graph from *sources* (nodes with $d_i > 0$) to *destinations* (nodes with $d_i < 0$), while for the remaining *transhipment nodes* (with $d_i = 0$) the total inbound flow must equal the total outbound flow. This problem has a huge set of applications, either in itself or, more often, as a submodel of more complex and demanding problems [1]. Without loss of generality we can restrict our analysis to connected graphs, as the general case of more than one connected component can be traced back to this case (basically, there is a separate LP for each one).

We study graph matrices coming from the application of IP methods, which have grown a well-established reputation as efficient algorithms for large-scale problems. In these methods, at each step we have to solve linear systems of the form

$$E\Theta E^T x = b \ , \tag{2}$$

where E is fixed, while $b \in \mathbb{R}^n$ and the $m \times m$ diagonal positive definite matrix Θ depend on the IP iteration; as we will not consider (possible) strategies for re-use of information between two different IP iterations, we will disregard this dependance, thus focussing our attention to the solution of (2) at any one fixed iteration. Since each diagonal element of Θ is associated to a specific arc of $\mathcal{G}$, we can consider $\mathcal{G}$ as a *weighted* graph, with Θ specifying the arc weights. In the following we will often use the shorthand $L = E\Theta E^T$, also disregarding the fact that L actually depends on Θ. As the following remark shows, L is closely tied to other well-known graph matrices.

Remark 2 *Let $\mathcal{G}' = (\mathcal{U}, \mathcal{V}')$ be the* undirected *graph obtained from $\mathcal{G}$ by ignoring the orientation of the arcs. $\mathcal{G}'$ also is a weighted graph, the weight w_{uv} of each edge $\{u, v\} \in \mathcal{V}'$ being the sum of the weights of all arcs of $\mathcal{G}$ which "collapse" in $\{u, v\}$ (note that multiple parallel arcs are allowed in (1), as they can have different cost and capacity). Then let $A(\mathcal{G}')$ be the symmetric (weighted) adjacency matrix of $\mathcal{G}'$, such that A_{uv} is the weight of the edge $\{u, v\}$ if it belongs to $\mathcal{V}'$ and 0 otherwise. Further let $D(\mathcal{G}')$ be the $n \times n$ the diagonal matrix with $D_{uu} = \sum_{v \in \mathcal{U}} A_{uv}$ (d_u is the node degree of u in the unweighted case). The* Laplacian *of $\mathcal{G}'$ is*

$$L(\mathcal{G}') = D(\mathcal{G}') - A(\mathcal{G}') \ ,$$

and it is easy to show that

$$L(\mathcal{G}') = E(\mathcal{G})\Theta(\mathcal{G})E(\mathcal{G})^T \ .$$

Hence, topological information about the original directed graph $\mathcal{G}$ is contained in $E(\mathcal{G})$ as well as $L(\mathcal{G}')$. The Laplacian of a graph has very many applications in such diverse fields as graph theory, statistics and combinatorial optimization [11, 24, 26].

In most general-purpose LP solvers, the linear systems (2) are solved by means of direct methods, typically the Cholesky decomposition preceded by a heuristic reordering of the columns of E aimed at minimizing the fill-in. For very large, sparse networks this approach is inefficient, as disastrous fill-in (e.g. [9]) may occur which renders the iteration cost unbearable. One thus has to revert to iterative methods instead, but these approaches can be competitive only if the rate of convergence is sufficiently high. This motivates studies of the extreme singular values of E and of the spectral behavior of L, since the convergence rate of iterative methods largely depends on the conditioning $\mu(\cdot)$ of the matrix [20]. In the first IP iterations, the matrix Θ is close to the identity and the spectral difficulties are mild: $\mu(L) \leq cn^2$, with c absolute constant and the bound attained up to lower order terms in the case of linear graphs [20]. However, in the last IP iterations the matrix Θ becomes highly unbalanced and the conditioning of L is essentially described by the wild conditioning of Θ. This phenomenon is analyzed in [29] for the case of linear graphs and this analysis is particularly relevant for the numerical solution of (2) through a preconditioned conjugate gradient (PCG) method. Most PCG-based IP algorithms make use of *subgraph* preconditioners of the form

$$L_{\mathcal{S}} = E_{\mathcal{S}}\Theta_{\mathcal{S}}E_{\mathcal{S}}^T$$

where $E_{\mathcal{S}}$ and $\Theta_{\mathcal{S}}$ denote the restriction of E and Θ, respectively, on the arcs of a "simple" subgraph $\mathcal{S}$ of $\mathcal{G}$, such as a spanning tree or some other triangulated graph [18, 19]. Computationally, spanning trees are the most effective choice in all but the most difficult cases, due to the fact that $\mathcal{S}$ can be chosen as an (approximate) Minimum Spanning Tree in $O(m)$, e.g. with the Prim algorithm.

PCG approaches using these preconditioners often work quite well, but there are some cases where the convergence rate is slow. The objective of this paper is therefore not to replace tree-based PCG approaches, but rather to complement them with ideas from the algebraic multigrid (AMG) field [32, 39]—and in particular from multi-iterative techniques [36]—to further improve their efficiency, in particular in the "difficult" cases.

2.2 Multi-iterative idea and MGM methods

Let $L \in \mathbb{C}^{n \times n}$ be a matrix with positive definite real part $\mathrm{Re}(L) = (L + L^*)/2$, $b \in \mathbb{C}^n$ be the right-hand-side and $l \in (0, n)$ (most often $l \approx \log n$) be the *number of levels*. Fix integers $n_0 = n > n_1 > n_2 > \ldots > n_l > 0$, take $R_{i+1}^i \in \mathbb{C}^{n_{i+1} \times n_i}$ full-rank matrices and consider a class $\mathcal{S}_i$ of iterative methods for n_i-dimensional linear systems. The related *V-cycle method* [40] produces the sequence $\{x^{(k)}\}_{k \in \mathbb{N}}$ according to the rule

$x^{(k+1)} = \mathcal{MGM}(0, x^{(k)}, b)$, with $\mathcal{MGM}$ recursively defined as follows:

$$
\begin{array}{c}
x_i^{(\mathrm{out})} := \mathcal{MGM}(i, x_i^{(\mathrm{in})}, b_i) \\
\hline
\begin{array}{rl}
\texttt{if(}\ i = l\ \texttt{)}\ \texttt{then} & \texttt{Solve}(L_l x_l^{(\mathtt{out})} = b_l) \\
\texttt{else} & \left|\begin{array}{ll}
\mathbf{1} & r_i \quad := L_i x_i^{(\mathrm{in})} - b_i \\
\mathbf{2} & b_{i+1} := R_{i+1}^i r_i \\
\mathbf{3} & L_{i+1} := R_{i+1}^i L_i (R_{i+1}^i)^H \\
\mathbf{4} & y_{i+1} := \mathcal{MGM}(i+1, 0_{n_{i+1}}, b_{i+1}) \\
\mathbf{5} & x_i^{(\mathrm{int})} := x_i^{(\mathrm{in})} - (R_{i+1}^i)^H y_{i+1} \\
\mathbf{6} & x_i^{(\mathrm{out})} := \mathcal{S}_i^{\nu}\left(x_i^{(\mathrm{int})}\right)
\end{array}\right.
\end{array}
\end{array}
\tag{3}
$$

Step 1 calculates the residual of the proposed solution. Steps 2, 3, 4, 5 define the *recursive coarse grid correction* by projection (step 2) of the residual, sub-grid correction (steps 3, 4) and interpolation (step 5), while step 6 performs some (ν) iterations of a "post-smoother".

By using the MGM as an iterative technique, at the k-th iteration, we obtain the linear systems $L_i x_i^{(k)} = b_i^{(k)}$, $i = 0, \ldots, l$, where the matrices $L_i \in \mathbb{C}^{n_i \times n_i}$ have all positive definite real part, i.e., $\mathrm{Re}(L_i) > 0$. Only the last one is solved exactly, while all the others are recursively managed by reduction to low-level system and smoothing. The procedures $\mathcal{S}_i$ are most often one-point methods [32] with prescribed linear part $S_i \in \mathbb{C}^{n_i \times n_i}$, that is,

$$
\mathcal{S}_i(x_i) = S_i x_i + (I_{n_i} - S_i) L_i^{-1} b_i^{(k)} \quad , \quad x_i \in \mathbb{C}^{n_i} \quad , \quad i = 0, \ldots, l-1 \ . \tag{4}
$$

If we recursively define the multigrid iteration matrix of level $i = l-1, \ldots, 0$ as

$$
\begin{cases}
MGM_l = O_{n_l \times n_l} \\
MGM_i = S_i^{\nu_i}\left[I_{n_i} - \left(R_{i+1}^i\right)^H \left(I_{n_{i+1}} - MGM_{i+1}\right) L_{i+1}^{-1} R_{i+1}^i L_i\right]
\end{cases} \quad , \tag{5}
$$

then $x_i^{(\mathrm{out})} = MGM_i x_i^{(\mathrm{in})} + (I_{n_i} - MGM_i) L_i^{-1} b_i$, so in the finer grid we have

$$
x^{(k+1)} = MGM_0 x^{(k)} + (I_{n_0} - MGM_0) L_i^{-1} b
$$

and MGM_i depends on i but not on any of the $x_i^{(k)}$ and $b_i^{(k)}$. For each $i = 0, \ldots, l-1$, the algorithm has essentially two degrees of indetermination:

- the choice of the projectors R_{i+1}^i;
- the choice of the smoothers $\mathcal{S}_i$.

The former, as well as the calculation of the L_i matrices, are performed before the beginning of the V-cycle procedure (pre-computing phase).

Two further modifications can also be applied:

- using a *pre-smoother*, i.e. adding a step 0 similar to step 6 where a further one-point method is employed;
- allowing, in steps 6 (and 0), the number of smoothing iterations to depend on the level i.

This corresponds to the matrix in (5) being multiplied on the right by a further iteration matrix, i.e.

$$
MGM_i = S_{i,\mathrm{post}}^{\nu_{i,\mathrm{post}}}\left[I_{n_i} - \left(R_{i+1}^i\right)^H \left(I_{n_{i+1}} - MGM_{i+1}\right) L_{i+1}^{-1} R_{i+1}^i L_i\right] S_{i,\mathrm{pre}}^{\nu_{i,\mathrm{pre}}} \quad ,
$$

where the number of smoothing steps $\nu_{i,\mathrm{pre}}$ and $\nu_{i,\mathrm{post}}$ depend on the level i. In practice (cf. [2, 37] and the references therein) the application of the pre-smoother accelerates the global convergence substantially, but the explanation of this phenomenon falls outside the convergence theory of the algebraic multigrid and

indeed pertains to multi-iterative methods [36]. For instance, looking just at the two-grid method, in the case of the d-dimensional discrete Laplacian, it is easy to prove that the post-smoothing given by the Richardson iteration with $\omega = \omega_1 \equiv 1/4d$ is strongly converging in the subspace of the high frequencies and that the coarse grid correction strongly reduces the error in the low frequencies subspace. Therefore, the combination of the two complementary iterations, which separately are slowly convergent on the global space $\mathbb{R}^n$, leads to a fast convergent two-grid method. However, a finer analysis tells us that the global error is now essentially localized in the middle frequencies: the iteration again given by Richardson but with with $\omega = \omega_2 \equiv 1/2d$ is not smoother, but it is fast convergent just in the middle frequencies subspace. Therefore, its further use in step 6 (or equivalently in step 0) increases very much the "spectral complementarity", so that we obtain a real multi-iterative method whose spectral radius is really small. We call an iteration having a spectral behavior complementary to both the coarse grid correction and the smoother an "intermediate iteration"; an example is the Richardson iteration with $\omega_2 = 2\omega_1$ in the case of the discrete Laplacian. As for the level-dependent number of smoothing iterations [37], it can be easily shown that a polynomial growth with i does not affect the global cost, that remains linear for banded or sparse structures, only changing the constants involved in the big O. This strategy, together with sophisticated preconditioners in the smoothing phases, was the key for developing very effective multigrid solvers for very ill-conditioned Sinc-Galerkin matrices [28]. However, while in the standard differential setting there is a gain in using an increasing $\nu_i = \nu_{i,\text{pre}} + \nu_{i,\text{post}}$, for problems with a smaller ill-conditioning (regularized Laplacian in [35]) the method that achieves the smallest theoretical cost and that minimizes the actual CPU times for reaching the solution with a preassigned accuracy is the simplest V-cycle with only one step of post-smoothing given by a classical damped Jacobi.

Since several possible choices exist in the implementation of the MGM approach, in the remainder of this section we will briefly discuss some initial computational tests, performed in the context of MCF problems, that provide a guidance about how to choose at least the two main components: projectors and smoothers.

2.3 Ghost node

Let $e = e_{(n)}$ be the all-ones vector of length n. It is immediate to realize that $e^T E = 0^T$ and therefore $e^T L = 0^T$, i.e., $\text{rank}(L) = n - 1$. However $e^T d = 0$ as well (for otherwise (1) cannot have any feasible solution [1]), and this property is transmitted to the right-hand sides b. Hence, by the Rouché-Capelli Theorem the linear system (2) has ∞^1 solutions of the form $x(\alpha) = \hat{x} + \alpha e$ for $\alpha \in \mathbb{R}$. Thus, let E' and b' be obtained by E and b (respectively) by erasing any one row and L' obtained by E' as usual. A solution to (2) can be obtained by solving the *cut system*

$$L'x' = b' \tag{6}$$

and setting $\hat{x} = (x', 0)$. Intuitively this amounts at creating a "ghost node" (the one corresponding to the deleted row) in the graph, as the columns of E corresponding to arcs entering (leaving) the ghost node will only have a 1 (-1) without the corresponding -1 (1).

It is well-known that, since L is a semidefinite positive matrix, CG or PCG approaches (started from the zero vector) solve (2) in the least-squares sense, i.e., find

$$\bar{x} = \underset{x \in \mathbb{R}^n}{\text{argmin}} \, \|Lx - b\|_2 \qquad \text{or equivalently} \qquad \|\bar{x}\|_2 = \min_{\alpha \in \mathbb{R}} \|\hat{x} + \alpha e\|_2 \ . \tag{7}$$

This is, in general, not true for other methods. Hence, using CG or PCG allows to work on the original graph at all levels but the last one, where eliminating one row (i.e. passing from (2) to (6)) to recover L'_l is necessary since a non-singular matrix is required by direct methods. Thus [P]CG can maintain our graph "sheltered from ghosts" which, as we will see, can have a positive impact on performances.

2.4 Smoothers

In accordance with the most successful strategies in the differential case, we tried as smoothers a combination of classical iterations. We recall that the discrete Laplacian can be viewed as a special instance of a graph matrix. In addition, by the analysis in [20], all the (unweighted) graph matrices with $\Theta = I$ share the

property that the small eigenvalues are related to smooth eigenvectors. In other words, as in the differential setting, the degenerating subspace is located essentially in the low frequencies. Therefore, we tested several combinations of pre- and post-smoothers—Gauss-Seidel with variation of relaxation parameter between 1/2 and 1, CG and PCG with elementary preconditioners—in a bit to find the one with better spectral complementarity and therefore performances. The preliminary numerical results were not encouraging (the only exception being a fairly good interaction between Gauss-Seidel and CG) especially in the last iterations of the IP process where the Θ becomes very unbalanced, with very large entries ($\approx$ `1e+6`) and very small entries ($\approx$ `1e-10`).

A similar occurrence was already experienced in [28], where the distribution of the nodes in the Sinc-Galerkin method induced very unbalanced diagonal entries. In that application, the only successful strategy was the combination of PCG smoothers with sophisticated and powerful preconditioners, in connection with the classical projection used in the differential context. Taking inspiration from this setting and from the similarity induced by the wild behavior of the diagonal entries, we focussed our attention on more powerful smoothers, that is the PCG family with a large set of specialized preconditioners:

- diagonal;
- incomplete Cholesky factorization (with zero fill-in);
- strongly incomplete Cholesky factorization;
- subgraph-based (in particular, tree-based) preconditioners.

However, as it will be clear from the next subsection, the problem with general graph matrices is more complicated than the one in [28], since the graph structure is not nearly as regular as those arising in the context of differential operators.

Concerning the number of smoothing steps, assume that the cost of MGM at the first level be $O(n) = \gamma n$ and consider the growth function $\nu_{i,\text{pre}} = i^k$. The total work required to perform a full MGM iteration is

$$W_l = \sum_{i=1}^{l} \gamma \frac{n}{2^{i-1}} i^k < \gamma n \sum_{i=1}^{\infty} \frac{i^k}{2^{i-1}} \quad . \tag{8}$$

The rightmost series in (8) is convergent; its initial values for $k = 0, 1, 2, 3, 4, 5$ are respectively 2, 4, 12, 52, 300 and 2164. This estimate has to be multiplied by 2 when employing both pre- and post-smoother, as in our case. We performed some preliminary numerical experiments with linear or a quadratic growth ($k \in \{1, 2\}$) on some of the most promising variants of smoothers, a sample of which is reported in Table 1 (see §6 for details about the numerical tests environment and the smoothers).

`GOTO IP 81`	**MGM$_2$**		**MGM$_3$**		**MGM$_{3b}$**		**MGM$_7$**		**MGM$_{7b}$**	
	it	it$_W$	it	it$_W$	it	it$_W$	it	it$_W$	it	it$_W$
$\nu_{i,\text{pre}} = \nu_{i,\text{post}} = 1$	75	75	9	9	14	14	9	9	9	9
$\nu_{i,\text{pre}} = \nu_{i,\text{post}} = i$	19	38	8	16	10	20	8	16	8	16
$\nu_{i,\text{pre}} = \nu_{i,\text{post}} = i^2$	12	72	8	48	8	48	8	48	8	48

Table 1: Number of MGM iterations in relation to the growth function

In the Table "it" denotes the number of MGM iterations, while "it$_W$" denotes the number of iterations taking into account the value of the rightmost series in (8). Most often the increased computational cost per multigrid iteration is not sufficiently compensated by the decrease in the number of iterations. Therefore in all the subsequent test we elected to use the simplest setting $k = 0$.

2.5 Projectors

Our initial interest was to verify whether standard operators with good performances in the context of differential equations, such as the very classical Full Weighting Operator (FWO)

$$R_{\text{FWO}} = \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 & & & & & & \\ & & 1 & 2 & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & 2 & 1 \end{pmatrix}, \tag{9}$$

would perform equally well in the context of general graph matrices. The preliminary results quickly made it clear that this was not the case, pushing us towards the development of the different operators described in this paper. In particular we quickly focussed our attention on Aggregation Operators (AGO) of the generic form

$$R_{\text{AGO}} = \begin{pmatrix} 1 & 1 & 1 & & & & & & \\ & & & 1 & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & 1 & 1 & 1 \end{pmatrix}, \tag{10}$$

which *preserve the graph structure* in the recursion, that is

$$R_{\text{AGO}} E \Theta E^T R_{\text{AGO}}^T \;=\; R_{\text{AGO}} L(\mathcal{G}) R_{\text{AGO}}^T \;=\; L(\mathcal{G}')$$

is the Laplacian of a new graph $\mathcal{G}'$ corresponding to the *aggregation of nodes* of the original graph (see Definition 2). Similar operators have been used with good results in different contexts, for example when designing multigrid solvers for Markov Chains [38].

A sample of the results of these preliminary experiments is provided in Table 2.5, where we report the number of iterations required for reaching a `1e-8` tolerance by a standard multigrid method (MGM), using Gauss-Seidel as pre-smoother and Conjugate Gradient (CG) as post-smoother (with $k = 0$, cf. §2.4), with two different choices of projectors: FWO and Aggregation Operators. In particular we use simple *blind* operators described in detail in §4.1. The experiments are performed on random graphs of different size (given in the header of each column), with random right-hand-side b. For comparison we also report results for three Krylov methods: CG and PCG with two different preconditioners (Diagonal and maximum Spanning Tree). We remark that random graphs have often been found to be "easy" for Krylov methods [18, 19] (`Net` problems are random, cf. §6.1), which is confirmed to be the case here. In order to evaluate the choice of the projectors in a "neutral" way with respect to the choice of the smoothers, we considered the case of graph matrix with balanced diagonal entries ($\Theta = I$, which corresponds to the first IP iteration).

RANDOM IP 1	**63**	**151**	**336**	**672**	**1024**
$\mathbf{MGM}_{\text{FWO}}$	15	23	73	99	–
$\mathbf{MGM}_{\text{AGO}}$	10	8	6	6	7
CG	31	24	19	15	53
$\mathbf{PCG}_{\text{D}}$	21	16	14	13	12
$\mathbf{PCG}_{\text{ST}}$	21	15	14	12	12

Table 2: Comparison of MGM with different projectors and other iterative approaches

The Table clearly shows that the linear interpolation (FWO) is not effective: in fact, the corresponding MGM is not efficient ("–" means a number of iterations higher than 100) even for these easy graphs and $\Theta = I$, which is generally the easiest case because it shows a moderate conditioning [20]. The results are also confirmed by the fact that using more sophisticated choices, such as quadratic interpolation [15] with stencil

$$\frac{1}{16}[1, 4, 6, 4, 1]$$

and even cubic interpolation [40] with stencil

$$\frac{1}{32}[-1, 0, 9, 16, 9, 0, -1]$$

does not lead to better results (actually, even to worse ones). Thus the differential setting is of no help in our graph problem. This is due to the fact that the matrix at lower levels is far away from a graph matrix and this destroys the structure which the multigrid method relies upon. Conversely the MGM approach with a simple structure-preserving projector appears to be competitive.

These tests suggests that, for general graphs and arc weightings (such as these of MCF matrices at final IP iterations), the only effective smoothers are sophisticated preconditioned Krylov methods. Therefore the only ingredient that may possibly make the V-cycle competitive is the right choice of the projectors. In other words, the key for having a fast method is the possibility of preserving the graph structure at all levels of the MGM approach. This is precisely what the rest of the paper is devoted to.

3 Graph operators

In this section we study the conditions under which projection operators preserve the graph structure of the matrix. We start with some preliminary definitions.

Definition 2 *An operator R is called a* graph *operator if, given any incidence matrix E,*

$$RE = E'\Theta' \ , \tag{11}$$

where E' is an incidence matrix and Θ' is a diagonal matrix.

It is evident that a graph operator preserves the graph structure at the lower levels, thus allowing a proper recursive strategy in the multigrid solver. A weaker notion is:

Definition 3 *An operator R is called a* graph operator for a given matrix *E if (11) holds for E, although it may not hold for all possible incidence matrices.*

Definition 4 *An operator R is called* admissible *if it does not have any column with all zero elements and any row with all equal elements.*

The rationale for this definition is that an all-zero column means that a node is "ignored", so that the projection ($r_i = (R^i_{i+1})^H r_{i+1}$) cannot produce any correction of the error. Symmetrically, a row with all equal elements in R means that RE has an all-zero row, i.e., E' in (11) has an isolated node. Furthermore note that AMG conditions impose that R has to be of full rank, so in any case a number for rows with all equal elements higher than one is not allowed. We remark that Definition 4 implies that $\mathrm{nnz}(R) \geq n$, where $\mathrm{nnz}(\cdot)$ denotes the number of non-zero elements. This leads to the following refinement of the concept:

Definition 5 *An admissible graph operator R such that* $\mathrm{nnz}(R) = n$ *is called* minimum.

Of course, any admissible operator with $nnz(R) > n$ is *non-minimum*, and the set of graph operators is partitioned between the two subsets. Similar definition can be given for graph operators w.r.t. a specific matrix E.

3.1 Necessary and sufficient conditions for graph operators

We now work our way towards a characterization of the set of graph operators. In this section we will stick to the following notation: $A^{[k]}$ is the k-th column of a generic matrix A, E is the $n \times m$ node-arc incidence matrix, for each $k = 1, \ldots, m$ the tail and head nodes of the corresponding arc (the row indices corresponding to 1 and -1 in $E^{[k]}$) are i_k and j_k, respectively, R is a $n' \times n$ operator with $n' < n$ (usually $n' \approx n/2$) so that $F = RE$ is a $n' \times m$ matrix, $e' = e_{(n')}$ is the all-ones vector of length n'.

Definition 6 *R is a* column operator *if the sum of elements of every column is constant, i.e., there exists $\rho \in \mathbb{R}$ such that $(e')^T R = \rho e^T$.*

Definition 7 *R is an* f-operator for E *if $(e')^T F = 0^T$. R is an* f-operator *if is is an f-operator for all graph matrices E.*

Theorem 3 *R is an f-operator for an incidence matrix E if and only if R is a column operator.*

Proof. [$\Rightarrow$] We want to prove that $r^T = (e')^T R = \rho e^T$ for some $\rho \in \mathbb{R}$. Suppose otherwise and let i, j be two indices such that $r_i \neq r_j$. Because $\mathcal{G}$ is connected, there exists at least one path in $\mathcal{G}$ having the nodes i and j as endpoints. It cannot be that $r_p = r_q$ for every arc $k = (p, q)$ belonging to the path, as this would imply $r_i = r_j$. Hence there must be $k = (p, q)$ such that $r_p \neq r_q$, which leads to

$$0 = (e')^T F^{[k]} = r^T E^{[k]} = r_p - r_q \neq 0 \quad :$$

a contradiction.
[$\Leftarrow$] For every column operator R and for every incidence matrix E we have $(e')^T F = \rho e^T E = 0^T$. ■

Corollary 4 *Let E be an incidence matrix. Any graph operator for E is a column operator.*

Proof. Since R is a graph operator for E, we have (cf. §2.3)

$$(e')^T F \;=\; (e')^T E' \Theta' = 0^T \quad ,$$

i.e., R is an f-operator for E. Hence by Theorem 3 it is a column operator. ■

Corollary 5 *Let R be a minimum operator. Then, with the possible exception of a scalar factor ρ, R is a binary matrix, i.e., $R \in \{0, 1\}^{n' \times n}$.*

Proof. By Definition 5 every column of the graph operator R only has one non-zero element and by Corollary 4 $(e')^T R = \rho e^T$, hence every non-zero entry must be equal to ρ. ■

Definition 8 *R is a* difference operator *for a set $\mathcal{I} = \{(i_1, j_1), (i_2, j_2), \ldots\}$ of pairs of node indices if for every $(i_k, j_k) \in \mathcal{I}$ the vector $R^{[i_k]} - R^{[j_k]}$ has either 0 or 2 non-zero elements.*

Lemma 6 *For each $k = 1, \ldots, m$, $F^{[k]} = R^{[i_k]} - R^{[j_k]}$.*

Proof. Immediate by the definition of E. ■

Theorem 7 *Let R be a column operator. R is a graph operator for E if and only if R is a difference operator for the set $\mathcal{I} = \mathcal{V}$ of the arcs of $\mathcal{G}$.*

Proof. [⇒] Assume by contradiction that R is not a difference operator for $\mathcal{I} = \mathcal{V}$. There exists an arc $k = (i_k, j_k) \in \mathcal{V}$ such that $R^{[i_k]} - R^{[j_k]}$ has neither 0 nor to 2 non-zero elements. This by Lemma 6 implies that the same holds for $F^{[k]}$, therefore R is not a graph operator, since (11) cannot hold.
[⇐] If R is a difference operator for $\mathcal{V}$, then for each $k = (i_k, j_k) \in \mathcal{V}$ the column $F^{[k]} = R^{[i_k]} - R^{[j_k]}$ has either 0 or 2 non-zeroes. Assume the latter and let p and q be the row indices of the non-zeroes in $F^{[k]}$. Since R is a column operator, by Theorem 3 we have

$$0 = (e')^T F^{[k]} = F_{pk} + F_{qk} \Rightarrow F_{pk} = -F_{qk} \ .$$

We can then scale $F^{[k]}$ by using the k-th diagonal entry of Θ', finally yielding $F = E'\Theta'$ where E' is an incidence matrix (possibly with "empty arcs"). Therefore R is a graph operator. ■

Remark 8 *It is easy to verify that minimum operators are columns operators and difference operators for any set $\mathcal{I}$.*

3.2 Admissible graph operator theorem

Finally, we prove a somewhat surprising result about admissible graph operators.

Definition 9 $C \in \mathbb{R}^{n' \times n}$ *is called a* row matrix *if each of its rows is a scalar multiple of e^T, i.e. $C = ce^T$ for some $c \in \mathbb{R}^{n'}$.*

Lemma 9 *Let $A \in \mathbb{R}^{n' \times n}$, $s \in \mathbb{R}^{n'}$ and $Q \in \mathbb{R}^{n' \times n}$ defined as*

$$Q_{ij} \ = \ \begin{cases} 0 & \text{if } A_{ij} = s_i \\ 1 & \text{otherwise} \end{cases} \ .$$

Then, for any pair (i, j) the number of non-zero elements of $A^{[i]} - A^{[j]}$ is equal to the number of non-zero elements of $Q^{[i]} - Q^{[j]}$ plus the number of indices k such that $A_{ki} \neq A_{kj}$ and $Q_{ki} = Q_{kj} = 1$.

Proof. For any k, denote $\alpha_k = A_{ki} - s_k$ and $\beta_k = A_{kj} - s_k$. If $Q_{ki} \neq Q_{kj}$, it means that $Q_{ki} = 1$ and $Q_{kj} = 0$ or vice versa; hence, $\alpha_k \neq 0$ and $\beta_k = 0$ or vice versa. Therefore $A_{ki} - A_{kj} = (A_{ki} - s_k) - (A_{kj} - s_k) = \alpha_k - \beta_k \neq 0$. When $Q_{ki} = Q_{kj} = 0$, one has $\alpha_k = \beta_k = 0$, and therefore $A_{ki} - A_{kj} = 0$. In the only remaining case $Q_{ki} = Q_{kj} = 1$ one directly counts whether $A_{ki} \neq A_{kj}$ or not, hence the proof is finished. ■

Theorem 10 *If R is an admissible graph operator and $n' > 2$ then*

$$R = M + C \ ,$$

where C is a row matrix and M is a minimum operator.

Proof. Since R is admissible, after a proper reordering of rows and columns of R, we have $R = [R_1 \mid R_2]$ where R_1 is a $n' \times n'$ matrix such that $\forall i \ R_{ii} \neq 0$. For minimum operators, without loss of generality, $R_1 = \rho \cdot I$, where ρ is a scalar factor and R_2 contains only copies of some columns of R_1.

We arbitrarily select one column of R—say $R^{[t]}$—as s and we apply Lemma 9 to R. Since R is admissible, there can't be an all-zeroes row in Q, because R does not have any row with all equal elements. Since R is also a graph operator, (11) holds for *any graph* $\mathcal{G}$. By Theorem 7 R is therefore a difference operator for *any set* $\mathcal{I}$, hence in particular for $\mathcal{I} = \{(1, t), (2, t), \ldots, (n, t)\}$ (the star tree rooted at t). By the definition of difference operator, in each column of Q there are either 0 or 2 ones; moreover, for any pair $Q^{[i]}$ and $Q^{[j]}$ of non-zero columns at least one of the two non-zeroes must be in the same row. In fact, assume by contradiction that $Q^{[i]}$ and $Q^{[j]}$ have all their non-zeroes in different positions: this means that there exists for Q a difference vector $Q^{[i]} - Q^{[j]}$ with 4 non-zero elements and by Lemma 9 the same holds for R, since there is no index k such that $Q_{ki} = Q_{kj} = 1$.

These two facts imply that, after a proper reordering of rows and columns of Q (which corresponds to a reordering of R), we have $Q = [Q_1 \mid Q_2]$ where Q_1 is the following $n' \times n'$ matrix

$$Q_1 = \begin{pmatrix} 0 & 1 & 1 & \cdots & 1 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{pmatrix}$$

and Q_2 contains only copies of some columns of Q_1. Now assume that R has in fact been reordered (if necessary) to match Q, consider the first row in (the reordered) R and pick any two column indices $h \neq j$ such that $Q^{[h]}$ and $Q^{[j]}$ are not all-zero; hence, $Q_{1h} = Q_{1j} = 1$ (remind that the columns in Q_2 are copies of these in Q_1, hence all the nonzero ones have a 1 in the first row). We claim that $R_{1h} = R_{1j}$. We start with the case where the other two nonzeroes are in different rows, i.e., $Q_{ph} = 1$ and $Q_{qj} = 1$ for $1 < p \neq q > 1$: by Lemma 9 we have a difference vector for R with 2 or 3 non-zero elements, depending on whether $R_{1h} = R_{1j}$ or not, which proves our claim since R is a difference operator. The case when $p = q$ easily follows by transitivity considering one further column h (which must exists) that has its other non-zero element in a different row: $R_{1i} = R_{1h}$ and $R_{1j} = R_{1h}$, so $R_{1i} = R_{1j}$. Note that the hypothesis $n' > 2$ is crucial in this part of the proof.

Consider now any row $i > 1$. Needless to say, all entries R_{ij} such that $Q_{ij} = 0$ have the same value (that of R_{ik}); therefore, we know that R has an arrangement like

$$R = [\, R_1 \mid R_2 \,] = \left(\begin{array}{ccccc|ccc} r_1 & c_1 & c_1 & \cdots & c_1 & c_1 & \ldots & c_1 \\ c_2 & r_2 & c_2 & \cdots & c_2 & r_{n'+1} & \ldots & c_2 \\ c_3 & c_3 & r_3 & \cdots & c_3 & c_3 & \ldots & c_3 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ c_{n'} & c_{n'} & c_{n'} & \cdots & r_{n'} & c_{n'} & \ldots & r_n \end{array}\right)$$

where the value $R_{ij} = r_j$ (possibly) $\neq c_i$ is at the unique row $i > 1$ (if any) such that $Q_{ij} = 1$. Consider any two column indices $h \neq j$ such that $Q_{ih} = Q_{ij} = 1$: we claim that, again, $R_{ih} = R_{ij}$, i.e., that all columns of R_2 are copies of some column of R_1 (the one having the nonzeroes in Q in the same position). This comes from the fact that, being a graph operator, R is also a column operator: $(e')^T R^{[j]} = (e')^T R^{[h]} = \rho$. Then, one has

$$\rho = (e')^T R^{[j]} = \sum_{p \neq i} c_p + r_j \qquad \text{and} \qquad \rho = (e')^T R^{[h]} = \sum_{p \neq i} c_p + r_h$$

whence $R_{ih} = r_h = \rho - \sum_{p \neq i} c_p = r_j = R_{ij}$. Again from the fact that R is a column operator, we have

$$\rho = (e')^T R^{[1]} = r_1 + c_2 + \sum_{p>2} c_p = (e')^T R^{[2]} = c_1 + r_2 + \sum_{p>2} c_p$$

and similarly for all pairs $(i, i+1)$ with $i < n'$, whence $r_1 - c_1 = r_2 - c_2 = \ldots = r_{n'} - c_{n'}$. Let us call the common value $r_i - c_i = \alpha$: it must be $\alpha \neq 0$, for otherwise one would have $r_i = c_i$ for all i, i.e., $R = C = ce^T$, contradicting the hypothesis that R is admissible (*each* row would have all equal elements). Hence, $M = R - C \neq 0$; in particular, M has exactly one nonzero per column, no all-zero rows, and all its nonzeroes have the same value α. Thus, M clearly is a minimum operator (cf. Corollary 5). ■

Remark 11 *The action on an incidence matrix E of any graph operator R and of the related minimum operator M (see Theorem 10) is the same. In fact, $RE = ME + CE = ME$.*

Remark 12 *The hypothesis $n' > 2$ in Theorem 10 is necessary, as for $n' = 2$ the thesis does not hold. In fact, the conditions which have to be satisfied are*

- *$R_{1j} + R_{2j} = \rho$ for all j*

- $R_{1j} = R_{1k}$ *if and only if* $R_{2j} = R_{2k}$ *for all* j *and* k

and these do not prevent from choosing all different R_{ij}*, so in general* $R \neq M + C$*. An easy counterexample with* $m = 5$ *is for instance*

$$R = \begin{pmatrix} 0.1 & 0.8 & -1 & 1 & 0.25 \\ 0.9 & 0.2 & 2 & 0 & 0.75 \end{pmatrix}$$

Corollary 13 *Let* R *be an admissible graph operator. If* R *is a binary matrix, then either* $R = M$ *or* $R = e'e^T - M$ *where* M *is a minimum operator.*

Proof. By Theorem 10 $R = M + C$. By Corollary 5 M is a binary matrix, with the possible exception of a scalar factor α. Hence either $C = 0$ and $R = M$ or C is the matrix with all entries equal to 1 and $R = C - M$ is the complement of the minimum operator M (the one having 0 where M has 1 and vice-versa). ■

We finish this section by noting that if the original graph $\mathcal{G}$ is connected (as is the case in our applications), then so is the "restricted" graph $\mathcal{G}'$.

Theorem 14 *Let* R *be an admissible graph operator; then,* $\mathcal{G}'$ *is a connected graph.*

Proof. By Remark 11 $RE = ME$, hence R acts on E as a minimum operator; these aggregate nodes, which can be adjacent (see §4.2) or not (see §4.1). Clearly, contracting nodes in an already connected graph produces a connected graph: each path in $\mathcal{G}$ corresponds to a (possibly non-simple) path in $\mathcal{G}'$ (in other words, connectivity in $\mathcal{G}'$ is the same as connectivity in the graph obtained by adding to $\mathcal{G}$ arcs forming cliques within the aggregated nodes). ■

3.3 Interpretation: the FWO operator for Poisson problems

The above results help explaining why the full weighting operator is a good choice for Poisson problems: basically, it is a graph operator for the corresponding very special graph matrix. Recalling its algebraic expression in (9) (and omitting the constant 1/4), one immediately notes that, leaving out the first and the last column, FWO is a column operator (with constant 2). The fundamental observation now is that the matrix associated with the Poisson problem has the EE^T form with the elimination of the first and the last row and the first and the last column):

$$(L_{\text{Poisson}})_n = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix}_{n \times n} = P_n E_n E_n^T P_n^T$$

$$\text{where} \qquad E_n = \begin{pmatrix} 1 & & & & \\ -1 & 1 & & & \\ & -1 & \ddots & & \\ & & \ddots & 1 & \\ & & & -1 & 1 \\ & & & & -1 \end{pmatrix}_{(n+2) \times (n+1)}, \; P_n = \begin{pmatrix} 0 \mid I_n \mid 0 \end{pmatrix}_{n \times (n+2)} .$$

The incidence matrix E is associated to the linear graph with $n + 2$ nodes, in which every node points to the next one, as depicted in Figure 1.

Indeed, R_{FWO}—which is clearly an admissible operator—is a difference operator for the set $\mathcal{I} = \{(2, 3), (3, 4), \ldots, (n - 2, n - 1)\}$, as the difference vector is

$$\begin{pmatrix} 2 \\ \\ \end{pmatrix} - \begin{pmatrix} 1 \\ 1 \\ \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ \end{pmatrix} \quad \text{or} \quad \begin{pmatrix} 1 \\ 1 \\ \end{pmatrix} - \begin{pmatrix} \\ 2 \\ \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ \end{pmatrix} .$$

Figure 1: Linear graph with $n + 2$ nodes. Nodes colored in grey are related to the Dirichlet boundary conditions.

The exceptions represented by the first and the last column, in which the difference vector has one non-zero element, do not contradict our results, as they can be explained by the elimination of rows and columns related to the Dirichlet boundary conditions. By Theorem 7 we conclude that R_{FWO} is a graph operator for the matrix E of Poisson problems (cf. Definition 3), although not a graph operator in general. Moreover, by Theorem 14 also the graph associated to $E' = E_{n'}$ is connected; better yet, it is still a linear graph. In fact,

$$R_{\text{FWO}}(L_{\text{Poisson}})_n P_{\text{FWO}} \;=\; \frac{1}{4}(L_{\text{Poisson}})_{n'} \;=\; \frac{1}{4} P_{n'} E_{n'} E_{n'}^T P_{n'}^T$$

where $P_{\text{FWO}} = 2(R_{\text{FWO}})^T$ is the operator of linear interpolation, $E_{n'}$ is the incidence matrix associated to the linear graph with $n'+2$ nodes, and $P_{n'}$ is defined as above. Thus the reason why FWO does work well in the Poisson case, but does not for general graphs, can be traced to the fact that it only preserves the graph structure for very simple (linear) graphs, but does not in the general case.

4 Minimum operators

As we saw in §3.2, the set of non-minimum operators is not empty, although it just contains row matrix "perturbations" of minimum operators. One may wonder whether non-minimum operators could be preferable to minimum ones. Remark 11 already suggests that this may be difficult to attain, since for the most part the effect of a non-minimum operator is similar to that of the corresponding minimum one. In practice, we did not identify any promising way to construct non-minimum operators; all our attempts with non-minimum operators invariably gave either equivalent or worse results than these with minimum ones. A small sample of our results are reported in Table 4 (see §6 for details on the MGM methods employed and the test instances), which reports the number of iterations required for reaching a `1e-5` tolerance with either $C = 0$ or randomly-generated $C = C(d, t)$, where d is density (percentage of non-zero rows of C) and t is the constant used for the (non-zero) values of c.

`GOTO IP 81`	**MGM$_2$**	**MGM$_3$**	**MGM$_{3b}$**	**MGM$_7$**	**MGM$_{7b}$**
$C = 0$	75	9	14	9	9
$C(5\%, -0.1)$	75	9	14	9	9
$C(5\%, -1/n)$	75	9	14	9	9
$C(5\%, +1/n)$	75	9	14	9	9
$C(5\%, +0.1)$	–	63	–	–	46
$C(30\%, -0.1)$	–	60	39	–	61
$C(30\%, -1/n)$	75	9	14	9	9
$C(30\%, +1/n)$	75	9	15	12	9
$C(30\%, +0.1)$	–	–	–	–	–

Table 3: Comparison minimum and non-minimum operators with different choices of C

As the Table shows, none of our choices of C provides any improvement in performances, which often significantly degrade. However the discussion in Section 3.3 shows that non-minimum operators can be very effective if the graph has a particular structure. Therefore, while it is still possible that further research may identify effective ways of choosing C to improve the performances, in the following we will concentrate on MGM approaches which use minimum operators only. These operators pick some subset of nodes and "shrink" them together into a super-node, which inherits all the incident arcs of the original ones, while arcs between any of the nodes shrank together disappear. Obviously a fundamental question has now to be answered, i.e., how to select the subset of nodes to be shrank.

There are many possible ways to approach this question. For instance, one possibly meaningful observation is that if $\mathcal{G}$ were a tree, then (2) could be solved in $O(n)$ [1]. More in general, if $\mathcal{G}$ were a triangulated graph then (2) could be solved in $O(m)$ [18, 19] (this is the idea subgraph-based preconditioners are based upon). A promising approach may be that of choosing the aggregation in such a way that after a few levels the "shrank" graph is triangulated, so as to pass as quickly as possible to the direct solution step. On the other hand, it is not clear how effective these aggregations may be in terms of reducing the overall number of MGM iterations. In the following we will describe some classes of minimum operators we have devised and tested to start shedding some light on this intricate issue. For each of them we describe the restriction operator $R \in \{0,1\}^{n' \times n}$, where n' is a fraction of n (n actually is n_k, i.e. the matrix dimension at the k-th level). The value $d = n - n'$ is called *descent parameter* and usually $d \approx n' \approx n/2$.

4.1 Blind projectors

We now present a first list of simple aggregation projectors. We call them *blind* operators, as their form is independent of the matrix to which they are applied. Blind operators are attractive from the computational viewpoint because they can be computed a-priori, therefore they are as cheap as they possibly can.

- The First Operator aggregates the first $d+1$ nodes into one:

$$R_{\text{First}} = \begin{pmatrix} 1 & 1 & \cdots & 1 & 1 & & & \\ & & & & & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix} .$$

- The Last Operator aggregates the last $d+1$ nodes into one:

$$R_{\text{Last}} = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & & & & & \\ & & & 1 & 1 & \cdots & 1 & 1 \end{pmatrix} .$$

- The Extreme Operator aggregates the first and the last $(d+1)/2$ nodes into one:

$$R_{\text{Extreme}} = \begin{pmatrix} 1 & 1 & 1 & & & & 1 & 1 \\ & & & 1 & & & & \\ & & & & \ddots & & & \\ & & & & & 1 & & \end{pmatrix} .$$

- The Medium Operator aggregates the $d+1$ "central" nodes into one:

$$R_{\text{Medium}} = \begin{pmatrix} 1 & & & & & & & \\ & \ddots & & & & & & \\ & & 1 & 1 & \cdots & 1 & & \\ & & & & & & \ddots & \\ & & & & & & & 1 \end{pmatrix} .$$

- The Random Operator is iteratively obtained by randomly selecting a set $\mathcal{R}$ of pairs of nodes (with $|\mathcal{R}| = d$) and aggregating them; formally, $R_{\text{Random}} = \prod_{(i,j)\in\mathcal{R}} R(i,j)$ where

$$R(i,j) = \begin{pmatrix} 1 & & i & & & j & & \\ & 1 & \downarrow & & & \downarrow & & \\ & & 1 & & & 1 & & \\ & & & \ddots & & & & \\ & & & & 1 & & & \\ & & & & & & 1 & \\ & & & & & & & 1 \end{pmatrix}$$

is the Pair Operator w.r.t. the pair (i, j).

- The Couple Operator aggregates every node with the following one:

$$R_{\text{Couple}} = \begin{pmatrix} 1 & 1 & & & & & \\ & & 1 & 1 & & & \\ & & & & \ddots & \ddots & \\ & & & & & 1 & 1 \end{pmatrix};$$

in other words, it is the product of $\approx n/2$ pair operators corresponding to pairs $(1, 2)$, $(3, 4)$, ...

Furthermore, it is clearly possible to alternate two or more of these operators along MGM levels.

Although these are the projectors used with success in §2.5 (at least when compared to the FWO), we found that the effectiveness of these operators is strongly influenced by the structure of the underlying graph $\mathcal{G}$. A sample of our results is shown in Table 4.1, where the results of the different blind projectors are compared on instances with rather different $\mathcal{G}$ and Θ (cf. again §6 for details).

	R_{First}	R_{Last}	R_{Extreme}	R_{Medium}	R_{Random}	R_{Couple}
MGM$_1$ – `GOTO IP 1`	34	34	36	27	31	10
MGM$_0$ – `NET IP 8`	38	16	23	28	13	19
MGM$_7$ – `GRID IP 32`	14	16	16	17	17	18

Table 4: Comparison of different blind projectors

The Table clearly show that the relative performances of the different blind projectors vary wildly as the underlying problem changes. For instance, for `GRID` at final IP iteration R_{First} is the best, while for `NET` at middle IP iteration is the worst. So it is difficult to choose any fixed blind projector whose performances are predictable and stable enough on a large class of instances. This suggests that it may be necessary to adapt the projector to the topological structure of the graph and maybe to the weights of the arcs too. That is, one must consider non-blind operators, despite the fact that they can more costly to determine.

4.2 Adaptive projectors

Several *adaptive* operators can be devised which depend on the values of the matrix at hand; since they may be more costly from the computational viewpoint, a non-trivial trade-off between increase of the performances and computational cost have to be struck. A simple idea to construct adaptive operators is to mimic the Random Operator (which, while not adaptive, is at least dynamic): select a set $\mathcal{R}$ of ($\approx n/2$) pairs of nodes and iteratively aggregate them. This is again equivalent to taking R as the product of the corresponding Pair Operators. Different operators then arise as a consequence of different rules to chose the elements of $\mathcal{R}$. In doing so, it is likely a wise choice to preserve as much as possible the topological structure of the graph.

Definition 10 *A graph operator R is a* contraction operator *for E if it aggregates exclusively adjacent nodes in $\mathcal{G}$.*

Remark 15 *R_{FWO} is a contraction operator for the matrix E of Poisson problem.*

Definition 11 *We say that* contraction property *holds for a class $\mathcal{C}$ of operators if, given any E, there exists an algorithm $\mathbf{A}$ which determines $\mathbf{A}(E) = R \in \mathcal{C}$ such that R is a contraction operator for E.*

Contractor operators are intuitively attractive, since they don't "mess up" with the structure of the graph. For instance, this is potentially useful when applying subgraph-based preconditioners, as discussed in §5.2. Because of this we have mostly concentrated on these while developing adaptive operators. In particular, a simple but effective scheme for choosing the next pair (i, j) to aggregate is the following:

- select i either at random or as the diagonal entry of L with the least (or the greatest) value;
- select j either at random or as the index of the non-zero off-diagonal entry in the i-th column with the least (or the greatest) *absolute* value (since $L_{ij} = -\Theta_{ij} < 0$).

When the process is repeated, the (iteratively) aggregated matrix $R(i,j)LR(i,j)^T$ is looked at instead of the original L. This gives rise to eight "$\{x,y\}$" projectors with x and y chosen between "rand", "min" and "max"; of course, the $\{\text{rand}, \text{rand}\}$ one being nothing else than R_{Random}. Our preliminary results showed that $\{x, \max\}$ projectors substantially outperform the corresponding variants where j is selected either at random or by looking at arcs with small weight; thus, as with subgraph-based preconditioners (cf. §5) choosing arcs with large weight for $\mathcal{R}$ appears to be the best option.

Proposition 16 *For the class of* $\{x, \max\}$ *operators, contraction property holds.*

Proof. $\{x, \max\}$ operators are minimum operators, so R is a graph operator for any E. The remaining essentially depends on the choice of j, which corresponds to a non-zero column element, and on Remark 2. ■

Regarding the choice of i, no clear winner emerges between "max" and "rand" uniformly in all IP iterations; however, the results are always quite similar, so the three strategies can be considered as equivalent. These the results seem to indicate that there could be still ample room for improvements; yet, they also very clearly indicate that already these simple adaptive projectors are much more dependable than blind ones. This is shown in Table 4.2 on a subset of instances for one of the most promising variant of MGM approach (see again §6 for details). The results are shown for both different classes of graphs and different sets of weights, corresponding to markedly different "regimes" in the IP algorithm. In particular, data is reported about the "opening game" first two iterations (where $\Theta \approx I$), the "end game" last iteration k and penultimate one $k-1$ (where Θ is highly unbalanced) and some "in the middle" ones.

$\mathbf{MGM_7}$	NET		GRID		GOTO	
IP iteration	BLI	ADA	BLI	ADA	BLI	ADA
1	5	5	5	5	6	11
2	6	6	12	12	–	12
$k/3$	7	7	8	9	–	39
$k/2$	–	11	8	8	–	22
$2k/3$	14	6	12	6	–	16
$k-1$	2	2	12	6	–	9
k	2	2	14	5	–	9

Table 5: Comparison of blind and adaptive projectors

As the Table shows, whereas blind (the best) approach (BLI) is competitive with an adaptive one (ADA) in some "easy" cases, the latter is the only option to reliably achieve acceptable performances.

It is clear that different, and possibly more sophisticated, choices of R may exist, even by restricting oneself to the class of aggregation operators obtained by the composition of a set $\mathcal{R}$ of two-nodes aggregations. For instance, every possible pair (i,j) actually defines the 2×2 minor

$$L^{ij} = \begin{bmatrix} L_{ii} & L_{ij} \\ L_{ji} & L_{jj} \end{bmatrix}$$

where $L_{ii} > 0$, $L_{jj} > 0$, $L_{ij} = L_{ji} < 0$ and $\det(L[i,j])$ is positive. Besides, because weak predominance for rows holds (if R is a graph operator, this is true at every MGM level), it is true that $|L_{ij}| \leq \min(L_{ii}, L_{jj})$. The choice of j of $\{x, \max\}$ operators implies that, if we suppose that $\min(L_{ii}, L_{jj}) = L_{jj}$, then $L_{ij} \approx L_{jj}$ and

$$\det(L[i,j]) = L_{ii}L_{jj} - L_{ij}^2 \approx (L_{ii} - L_{jj})L_{jj} \ ,$$

i.e., if $L_{ii} \approx L_{jj}$ then one aggregates a badly conditioned part, whereas if $L_{ii} \gg L_{jj}$ then one aggregates a nicely conditioned part. Hence may use quantities related to $\det(L[i,j])$, such as "normalized" versions like $\det(L[i,j])/(L_{ii}^2+L_{jj}^2)$ or $\det(L[i,j])/(L_{ii}L_{jj})$, which measures how well or badly conditioned the 2×2 minor is, to gauge how promising a (i,j) pair is. This gives rise to max-minor or min-minor operators, depending on whether one chooses to preferentially aggregate well-conditioned or ill-conditioned minors.

Our experience with these two variants is that aggregating badly conditioned minors is by far the most effective variant. The choice of the (i,j) pair is done as in the previous case: first i is selected with either one of the three above strategies, then j is selected in $O(n)$ ($O(1)$ for sparse graphs) as the one giving the most ill-conditioned minor. Note that it may be possible to determine the "overall best" minor operator by looking at all arcs in $\mathcal{G}$, but that would have a $O(m)$ cost. The $\{x, \text{min-minor}\}$ operator seems to somewhat improve upon the previous $\{x, \max\}$ ones; in particular, its performances seems to be even less dependent by the choice of i, so the average behavior is somewhat better. This is especially true when treating extremely ill-conditioned problems, which therefore makes it our adaptive projector of choice.

Clearly even more complex adaptive approaches may be devised. For instance, one may compute an arc weight for each (i,j) basedon the above measure of conditioning of $L[i,j]$, and then solve a *min-cost matching problem* to determine the "best" set $\mathcal{R}$ of node-disjoint arcs with respect to the given weights. This may lead to better "global" choices of $\mathcal{R}$ with respect to these obtained by a sequence of locally optimal choices, but exact solution of matching problems typically costs much in excess to $O(nm)$ [1] which for practical purposes may already be too much. Achieving a proper trade-off between the computational cost for defining an effective operator and the improvement in convergence speed and cost per iteration of the resulting MGM is non-trivial. For instance, one may force somewhat a bipartite structure on the graph (e.g. by purposely ignoring the joint effect of multiple aggregations concerning the same node) and resort to fast randomized and/or approximate algorithms for *bipartite matching* problems, which require a significantly lower effort [34]. Conversely one may want to explicitly measure the effect of multi-node aggregation moves, which leads to more general *clustering* problems, so further theoretical and computational investigation on the matter is required. However, since the results of simple adaptive operators are already compelling enough, this is left for future developments.

5 Relationships with preconditioning

As anticipated in Section 2.4 and numerically illustrated in the next one, PCG with sophisticated subgraph-based preconditioners is the most promising class of approaches to serve as effective pre- and post-smoothers in MGM methods. Thus, at least by restricting to *contraction operators* R, one actually have to choose *two subsets of arcs* at each level of the MGM: $\mathcal{S}$ for the preconditioner, $\mathcal{R}$ for the projector.

Clearly the two choices are not entirely independent. In particular $\mathcal{R}$ at one level influences the graph and therefore possibly $\mathcal{S}$, at the next. We therefore aim at exploring more in detail the relationships between the two choices; to this aim, we now present and analyze two possible preconditioning techniques which takes into account the effect of projection. In the following, we denote by $S = E_{\mathcal{S}}\Theta_{\mathcal{S}}E_{\mathcal{S}}^T$ the subgraph-based preconditioner. We assume S positive definite and in fact "easy" to invert, which requires specific care in the choice of $\mathcal{S}$ [18, 19].

5.1 Inverse projection

The preconditioned matrix of PCG at the first level of the MGM approach is $P = S^{-1}L$. At the k-th level, *inverse projection* uses the preconditioned matrix

$$P_k = (R_kR_k^T)^{-1}R_kS^{-1}R_k^T(R_kR_k^T)^{-1}R_kLR_k^T \tag{12}$$

where R_k is the cumulative restriction operator after k levels. The analysis of inverse projection requires the following two lemmas, whose proofs are based on continuity and density arguments of invertible matrices in the space of all matrices (see e.g. [6]).

Lemma 17 *For all $A, B \in \mathbb{C}^{n\times n}$ the characteristic polynomials of AB and BA coincide and therefore $\sigma(AB) = \sigma(BA)$.*

Lemma 18 *Let $A \in \mathbb{C}^{m\times n}$ and $B \in \mathbb{C}^{n\times m}$ with $m > n$, then for the characteristic polynomials of AB and BA one has $p_{AB}(\lambda) = p_{BA}(\lambda) \cdot \lambda^{m-n}$ and therefore $\sigma(AB) = \sigma(BA) \cup \{0, \ldots, 0\}$ (repeated $m-n$ times).*

Theorem 19 *If $\sigma(P) \subset [a, b]$ then $\sigma(P_k) \subset [a_k, b_k]$ with $a_k > a$.*

Proof. We denote by $(a \leq)\ \alpha_1 \leq \ldots \leq \alpha_n\ (\leq b)$ the eigenvalues of P and with $\beta_1 \leq \ldots \leq \beta_{n_k}$ those of P_k. We single out $Q_k = R_k^T(R_kR_k^T)^{-1}R_k$ from (12). Thanks to Lemma 18, Q_k has only 0 and 1 eigenvalues (as the non-zero ones are the same of $R_kR_k^T(R_kR_k^T)^{-1} = I$). Again by Lemma 18, P_k has essentially the same spectrum as $S^{-1}Q_kLQ_k$, which in turn is similar to $S^{-1/2}Q_kLQ_kS^{-1/2}$. In other words, its eigenvalues are

$$0 = \lambda_1 = \lambda_2 = \ldots = \lambda_{n-n_k} < \lambda_{n-n_k+1} \leq \ldots \leq \lambda_n$$

where $\lambda_{n-i} = \beta_{n_k-i}$ for $i = 0, \ldots, n_k - 1$. We can now apply the MinMax Theorem to "discover" the first non-zero eigenvalue ($\lambda_{n-n_k+1} = \beta_1$):

$$\begin{aligned}\lambda_{n-n_k+1} &= \min_{\dim(U)=n-n_k+1} \max_{u\in U} \frac{u^TS^{-1/2}Q_kLQ_kS^{-1/2}u}{u^Tu} \\ &= \min_{\dim(V)=n-n_k+1} \max_{v\in V} \frac{v^TQ_kLQ_kv}{v^TSv} \qquad \text{, where} \quad v = S^{-1/2}u\ .\end{aligned}$$

Now let X be the n_k-dimensional space generated by columns of R_k^T. For every $x \in X$ we have $Q_kx = [R_k^T(R_kR_k^T)^{-1}R_k]R_k^T\tilde{x} = R_k^T\tilde{x} = x$. On the other hand, for every $x \in X^\perp$ we have $Q_kx = 0$. Since V is a $(n - n_k + 1)$-dimensional space and X is a n_k-dimensional one, their intersection $Z = V \cap X$ must have dimension at least 1. Therefore, taken $z \in Z$, we conclude

$$\lambda_{n-n_k+1} \geq \min_{\dim(V)=n-n_k+1} \frac{z^TQ_kLQ_kz}{z^TSz} = \min_{\dim(V)=n-n_k+1} \frac{z^TLz}{z^TSz} \geq \alpha_1 \geq a\ .$$

■

Had the result been $[a_k, b_k] \subset [a, b]$, one would have reached the very significant conclusion that the spectral properties of the preconditioned matrix get better and better as the level increases, so have good hopes that the same holds for the practical behavior of the PCG. Theorem 19 instead only gives indications about the lower extreme of the interval; however, this is the most important one for PCG convergence, as well emphasized in [3]. Therefore, it should be expected that a good practical behavior of PCG be observed at all levels of the MGM approach, provided that $\mathcal{S}$ has been properly chosen at the first level.

5.2 Dense projection

A different (and somewhat simpler) approach is the *dense projection*, that entails the use of

$$S_k = R_kSR_k^T$$

as preconditioner at the k-th level. The advantage is that S_k is the subgraph-based preconditioner on the graph at the k-th level simply obtained by applying to $\mathcal{S}$ the same aggregations applied to $\mathcal{G}$, so it is very inexpensive to compute. However, this is dubbed dense since number of non-zero elements can significantly increase with respect to that of the original subgraph $\mathcal{S}$ (triangulated subgraphs are typically fairly sparse), thus causing a relevant growth in the cost for inverting S_k. It is possible to choose $\mathcal{R}$ appropriately so as to avoid this issue.

Theorem 20 *Let $\mathcal{S}$ be a triangulated graph, $(i, j) \in \mathcal{S}$ and $\mathcal{S}'$ obtained by $\mathcal{S}$ by aggregating i and j. Then $\mathcal{S}'$ is a triangulated graph.*

Proof. Basically it all depends on the fact that aggregating an existing arc cannot create any new cycle. Assume by contradiction that a cycle $\mathcal{C}'$ of length $k \geq 4$ exists in $\mathcal{S}'$ which has no *chord* (an arc joining two non-consecutive vertices in $\mathcal{C}'$), thus negating triangularity of $\mathcal{S}'$. Hence $\mathcal{S}'$ and $\mathcal{S}$ are as depicted in Figure 2, where continuous lines indicate arcs which are surely present, whereas *at least* one arc of pairs indicated with dotted and dashed lines is present; in plain words, p and q are not adjacent in $\mathcal{S}$, but there exists a path joining them (passing through i or j). Let $\mathcal{Y}$ be the subgraph $\mathcal{C}' \setminus \{i'\}$, which is a linear graph of order $k-1$ belonging both to $\mathcal{S}'$ and $\mathcal{S}$: neither i nor j are adjacent to nodes of $\mathcal{Y} \setminus \{p, q\}$. Therefore, there exists in $\mathcal{S}$ a cordless cycle $\mathcal{C} = \mathcal{Y} \cup \{i\}$ (or $\mathcal{Y} \cup \{j\}$) of length k, and this concludes the proof.

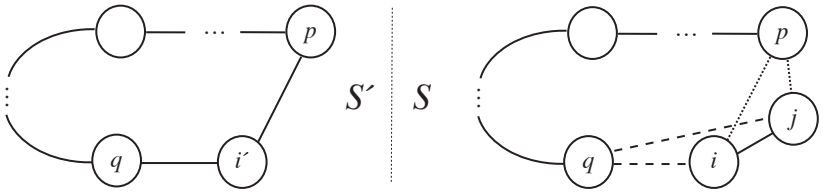

Figure 2: Aggregation in a triangulated graph preserves the property

■

Thus, choosing $\mathcal{R} \subset \mathcal{S}$ ensures avoidance of any fill-in effect. It is clear that such a choice severely restricts the set of available projection operators, possibly unnecessarily so. For instance, it is easy to realize that if $\mathcal{S}$ is a tree, then triangularity is preserved by R_{ij} not only if $(i, j) \in \mathcal{S}$ (joining "a father and a son"), but also if i and j are "brothers", i.e. they have a common adjacent node in $\mathcal{S}$. Allowing to aggregate along arcs in $\mathcal{V} \setminus \mathcal{S}$ may be possible without incurring in fill-in and any improvement in the flexibility of the approach is in principle desirable.

Exploiting arcs "joining brothers" is precisely the idea behind *Brother-Connected Trees*, the most effective more-than-tree subgraph-based preconditioners [18, 19]. These are, roughly speaking, obtained by adding this kind of arcs to an existing (or being constructed) spanning tree. In other words, if $\mathcal{S}$ were a tree and $(i, j) \notin \mathcal{S}$ were an arc joining brothers which may be deemed useful to be part of $\mathcal{R}$, then (i, j) could have been added to $\mathcal{S}$ in the first place. It appears that more-than-tree triangulated preconditioners may be even more attractive in the context of MGM methods (at least when using minimum operators and dense projection) than they are in the context of direct application of PCG, since adding arcs to $\mathcal{S}$ not only improves its preconditioning capabilities, but also improves the set of available choices for the projector. Then again, finding the appropriate balance between the increase in computational cost due to finding and factoring a larger preconditioner and the corresponding decrease in iterations count is already rather delicate in the PCG context, even more so in the MGM one. Therefore in our experiments we have limited ourselves to simpler tree-based preconditioners, leaving more complex schemes for future research.

Both inverse projection and dense projection approaches seem to hold promises for finding the right balance between the selection of the preconditioner and that of the projector. Choosing $\mathcal{R}$ and $\mathcal{S}$ so as to attain good convergence results at all levels with an acceptable computational cost remains a significant challenge, which will require further investigation. The theoretical and experimental investigation performed in this paper will provide a sound starting basis for such developments.

6 Numerical tests

In this section we report a wide set of numerical tests aimed at exploring the computational behavior of MGM approaches on instances of (2) coming from real applications, in particular MCF problems. Having (as far as the scope of the present paper permits) settled the issue of the choice of the projector R in the discussion of the previous sections, here we concentrate on the other fundamental ingredient of a successful MGM approach, i.e. the appropriate selection of pre- and post-smoother.

6.1 Problem generators

The tests have been performed on matrices L coming from the solution of randomly-generated MCF instances. Three different well-known random problem generators have been used: `Net`, `Grid`, `Goto`. Each generator produces matrices with different topological properties, as shown in Table 6.1. Furthermore, the solution of the MCF instances via an IP methods produces weight matrices Θ with a different behavior.

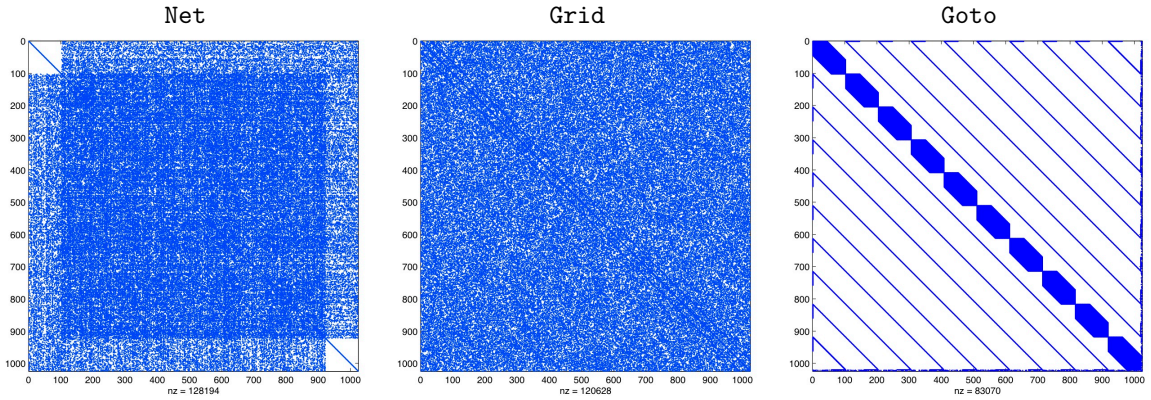

Table 6: Structure of L for different problem classes

As the pictures show, the graph in `Net` problems has a random topological structure; these are the easiest instances to solve with the IP algorithm. Both `Grid` and `Goto` (Grid On TOrus) problems have a grid structure, but the latter are considerably more difficult to solve than the former, both in terms of IP algorithm and as the corresponding linear system. The difficulty of `Goto` is likely to be related by the the structure of the L matrix, which is far from the block and the banded case; we recall that the latter is the classical pattern related to standard grid graphs. Under the same conditions (problem size, IP iteration and preconditioning), generally a `Goto` system requires an order of magnitude more PCG iterations than `Net` or `Grid` ones.

6.2 List of methods

For our experiments, we have compared several variants of MGM with a few reference Krylov approaches: a standard Conjugate Gradient methods (**CG**) and four Preconditioned Conjugate Gradient methods differing only for the preconditioning scheme. The preconditioners we tested are diagonal ($\mathbf{PCG_1}$), strongly incomplete Cholesky ($\mathbf{PCG_2}$), zero fill-in incomplete Cholesky ($\mathbf{PCG_{2b}}$) and spanning tree ($\mathbf{PCG_3}$). The details of the 11 tested versions of MGM approach are reported in Table 6.2. "Inverse projection" and "dense projection" refer to PCG approaches where the preconditioning schemes are those described in §5.

6.3 Results

All methods showed convergence, which however in several cases was exceedingly slow; thus, we had to resort to setting an a priori upper-bound for the number of iterations. This was (somewhat arbitrarily) chosen as 5000 iterations for Krylov methods and at 100 iterations for MGM one, considering that the latter have a higher cost per iteration. Tables 6.3, 6.3 and 6.3 report the number of iterations required to achieve `1e-5` accuracy on instances with $n = 1024$ and $m = 65535$ for `Net`, `Grid` and `Goto` instances, respectively. Each column report results for the same matrix at different IP iterations, comprising "opening game", "middle" and "end game" ones (k denotes the last IP iteration, cf. §4.2). Methods requiring more than the maximum allotted number of iterations are denoted by $*$ and $-$.

The results confirm that subgraph-based preconditioners are by far the most effective Krylov approach (among the tested ones), in particular in the "difficult" last IP iterations where Θ is highly unbalanced.

id	pre-smoother			post-smoother		
level	first	middle	last	first	middle	last
MGM$_0$	Gauss-Seidel			CG		
MGM$_1$	**PCG$_1$**			**PCG$_1$**		
MGM$_2$	**PCG$_3$**	**PCG$_1$**		**PCG$_3$**	**PCG$_1$**	
MGM$_3$	**PCG$_3$**	**PCG$_2$**	**PCG$_3$**	**PCG$_3$**	**PCG$_2$**	**PCG$_3$**
MGM$_{3b}$	**PCG$_3$**	**PCG$_{2b}$**	**PCG$_3$**	**PCG$_3$**	**PCG$_{2b}$**	**PCG$_3$**
MGM$_4$	**PCG$_2$**			**PCG$_2$**		
MGM$_{4b}$	**PCG$_{2b}$**			**PCG$_{2b}$**		
MGM$_5$	**PCG$_3$**	inverse projection		**PCG$_3$**	inverse projection	
MGM$_6$	**PCG$_3$**	dense projection		**PCG$_3$**	dense projection	
MGM$_7$	**PCG$_3$**			**PCG$_3$**		
MGM$_{7b}$	**PCG$_3$**	**PCG$_2$**		**PCG$_3$**	**PCG$_2$**	

Table 7: List of MGM methods

id	1	2	$k/3$	$k/2$	$2k/3$	$k-1$	k
CG	41	51	63	1433	*	*	*
PCG$_1$	8	16	28	140	1213	4789	2431
PCG$_2$	8	7	8	17	109	473	221
PCG$_{2b}$	5	7	7	35	282	1172	652
PCG$_3$	8	10	11	21	14	6	3
MGM$_0$	8	9	13	–	–	–	–
MGM$_1$	5	10	13	–	–	–	–
MGM$_2$	5	6	6	9	8	2	2
MGM$_3$	5	6	7	11	7	2	2
MGM$_{3b}$	5	6	7	11	8	2	2
MGM$_4$	5	4	4	8	63	–	–
MGM$_{4b}$	5	4	4	8	–	–	–
MGM$_5$	5	6	6	11	10	2	2
MGM$_6$	5	6	6	11	6	2	2
MGM$_7$	5	6	7	11	6	2	2
MGM$_{7b}$	5	6	7	11	7	2	2

Table 8: Results for `NET` instances

Cholesky-based preconditioners are competitive in the first IP iterations, but they are consistently outperformed in the last ones. It is therefore not surprising that the best MGM (especially in the difficult cases) are those which make use – all or in part – of tree preconditioners, such as **MGM$_3$**, **MGM$_{3b}$**, **MGM$_7$** and **MGM$_{7b}$**. Again Cholesky-based PCG smoothing can be competitive in the initial IP iterations, but they are quickly (even very quickly, in the "difficult" `Goto` instances) outgunned.

For **MGM$_0$** holds a different speech from the other MGMs, because one iteration of this method is less expensive than one iteration of a sophisticated PCG. Therefore in the first IP iterations of all problems we have a MGM that is competitive compared with PCG techniques.We remark that this also crucially depends on an appropriate choice of projector, as the performances of the simple MGM are strongly influenced by R. Conversely MGM approaches with more powerful preconditioners are much more resilient to suboptimal choices of the projector.

An interesting performance measure to gauge the effectiveness of MGM approach is

$$\boldsymbol{\theta}_p = \frac{\text{number of steps of the best } \mathbf{PCG} \text{ at IP iteration } p}{\text{number of steps of the best } \mathbf{MGM} \text{ at IP iteration } p}$$

which is reported in Table 6.3. Since $\nu_i = 2$ (one pre-smoother and one post-smoother), it is reasonable to

id	1	2	$k/3$	$k/2$	$2k/3$	$k-1$	k
CG	17	25	22	75	*	*	*
$\mathbf{PCG_1}$	13	11	13	39	740	2174	2071
$\mathbf{PCG_2}$	12	9	9	12	73	146	148
$\mathbf{PCG_{2b}}$	7	6	7	17	233	593	576
$\mathbf{PCG_3}$	12	11	13	20	27	9	10
$\mathbf{MGM_0}$	5	19	9	15	–	–	–
$\mathbf{MGM_1}$	5	12	9	23	–	–	–
$\mathbf{MGM_2}$	5	12	9	8	10	9	7
$\mathbf{MGM_3}$	5	11	9	8	5	7	6
$\mathbf{MGM_{3b}}$	5	11	8	8	8	7	6
$\mathbf{MGM_4}$	5	9	7	4	10	53	–
$\mathbf{MGM_{4b}}$	5	9	7	4	25	–	–
$\mathbf{MGM_5}$	5	12	9	8	6	6	6
$\mathbf{MGM_6}$	5	12	9	8	6	6	5
$\mathbf{MGM_7}$	5	12	9	8	6	6	5
$\mathbf{MGM_{7b}}$	5	11	9	8	8	5	6

Table 9: Results for `GRID` instances

compare performances by iteration steps, considering that the MGM cost will be at least twice—and likely around four times, cf. §2.4—that of a PCG iterations with analogous preconditioning.
A first observation is that, with only one exception, all entries are at least as large as 1, meaning that—at least in terms of iterations count—the best MGM is always faster than the best PCG. However, taking into account the difference in iterations cost, MGM may not be strikingly competitive with PCG on "easy instances". The case it is radically different for the "difficult" `Goto` problems: indeed very early on (in terms of IP iterations) $\boldsymbol{\theta}_p$ gets larger than 4 and it can get above 40. Therefore the much higher speed of convergence of MGM surely renders it highly competitive in this case. Moreover, in problems in which we can find a particularly effective projection, like in case of some tested `Goto` problems, MGM shows a substantially higher speed of convergence when compared with the best PCG technique and hence a competitive total cost. The latter fact represents a precious indication which is worth studying in depth.

7 Conclusions

We have considered multi-iterative techniques of multigrid type for the numerical solution of large linear systems with (weighted) structure of graph. We combined proper coarser-grid operators with auxiliary techniques, by showing that the most effective smoothers typically are Krylov type with subgraph-based preconditioners, while the projectors have to be designed for maintaining as much as possible the structure of graph matrix at the inner levels. Some necessary and sufficient conditions have been proved; interestingly enough, this framework is useful for explaining the reason why the classical projectors inherited from differential equations are good in the differential context and why they behave unsatisfactorily for unstructured graphs. Several numerical experiments have been conducted showing that our approach is effective especially in very difficult cases, where even the best known preconditioned Krylov approaches are rather slow.

Several research lines seems to be worth future investigation. From the theoretical viewpoint, it would be very interesting to obtain a rigorous characterization of the convergence speed of the proposed multigrid techniques with varying pre- and post-smoothers and choices of the projector. Regarding the latter, a better understanding of the effect of the projection step on the spectral properties of the matrices at lower level would be required to design more effective approaches to choosing R, possibly simultaneosuly taking into account selection of an appropriate subgraph for preconditioning techniques. From the practical viewpoint, one particularly appealing (and challenging) application arises if the matrix B of the Google problem (cf. Remark 2)

$$Bx = x \qquad , \qquad x \in \mathbb{R}^n_+ \qquad , \qquad \|x\|_1 = 1$$

id	1	2	$k/3$	$k/2$	$2k/3$	$k-1$	k
CG	27	917	$*$	$*$	$*$	$*$	$*$
$\mathbf{PCG_1}$	17	270	1956	$*$	$*$	$*$	$*$
$\mathbf{PCG_2}$	15	35	311	1676	$*$	$*$	$*$
$\mathbf{PCG_{2b}}$	8	39	614	1359	2848	$*$	4443
$\mathbf{PCG_3}$	16	55	183	342	455	405	385
$\mathbf{MGM_0}$	17	90	–	–	–	–	–
$\mathbf{MGM_1}$	10	70	–	–	–	–	–
$\mathbf{MGM_2}$	8	15	47	54	73	66	75
$\mathbf{MGM_3}$	6	11	44	15	10	9	9
$\mathbf{MGM_{3b}}$	5	12	45	21	15	14	14
$\mathbf{MGM_4}$	6	11	75	–	–	–	–
$\mathbf{MGM_{4b}}$	6	13	66	–	–	–	–
$\mathbf{MGM_5}$	10	23	–	–	–	–	–
$\mathbf{MGM_6}$	6	15	48	98	–	–	–
$\mathbf{MGM_7}$	6	12	39	22	16	9	9
$\mathbf{MGM_{7b}}$	6	10	39	15	21	9	9

Table 10: Results for GOTO instances

$\boldsymbol{\theta}_p$	NET	GRID	GOTO
1	1.00	1.40	1.60
2	1.75	0.67	3.50
$k/3$	1.75	1.00	4.69
$k/2$	2.13	3.00	22.80
$2k/3$	2.33	5.40	45.50
$k-1$	3.00	1.80	45.00
k	1.50	2.00	42.78

Table 11: Relative efficiency of best PCG vs. best MGM

is symmetric; in fact, since $D = I$ in Google, one has that

$$L = I - B = E\Theta E^T$$

and therefore hence our techniques can be applied to

$$(E\Theta E^T)x = 0 \ .$$

Thus, it would be very interesting to test the graph preserving projectors devised in this paper on linear systems arising in the Google setting, considering that the smoothers could be chosen among the various (preconditioned) Krylov iterative solvers already discussed and tested in [14].

Acknowledgements

This work was supported by Università degli Studi dell'Insubria under a grant for research activity. We are grateful to Claudio Gentile and Marco Donatelli for useful discussion and support.

References

[1] R.K. Ahuja, T.L. Magnanti and J.B. Orlin. *Network flows: theory, algorithms and applications.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] A. Aricó, M. Donatelli and S. Serra-Capizzano *V-cycle optimal convergence for certain (multilevel) structured linearsystems.* Siam J. Matrix Anal. Appl., 26-1 (2004), pp. 186–214.

[3] O. Axelsson, G. Lindskög. *The rate of convergence of the preconditioned conjugate gradient method.* Numer. Math., 52 (1986), pp. 499–523.

[4] O. Axelsson, M. Neytcheva. *The algebraic multilevel iteration methods – theory and applications.* Proc. of the 2nd Int. Coll. on Numerical Analysis, D. Bainov Ed., Plovdiv (Bulgaria), 1993, pp. 13–23.

[5] M.S. Bazaraa, J.J. Jarvis and H.D. Sherali. *Linear programming and network flows.* Wiley, New York, NY, 1990.

[6] R. Bhatia. *Matrix Analysis.* Springer Verlag, New York, 1997.

[7] E.G. Boman, D. Chen, B. Hendrickson and S. Toledo. *Maximum-weight-basis preconditioners.* Numer. Linear Algebra Appl. 11(2004), no. 8-9, pp. 695–721.

[8] E.G. Boman, B. Hendrickson. *Support theory for preconditioning.* Siam J. Matrix Anal. Appl. 25 (2003), no. 3, 694–717.

[9] J. Castro *A specialized interior-point algorithm for multicommodity network flows.* Siam J. Optim., 10 (2000), pp. 852–877.

[10] J. Castro, A. Frangioni. *A parallel implementation of an interior-point algorithm for multicommodity network flows.* in Vector and Parallel Processing – VECPAR 2000, J.M. Palma, J. Dongarra and V. Hernandez eds., Lecture Notes in Computer Science Vol. 1981 (2001), Springer-Verlag, p. 301–315.

[11] A. Cayley. *A theorem on trees.* Quart. J. Math. 23 (1889) 376–378.

[12] D. Cherubini, A. Fanni, A. Frangioni and A. Mereu. *Primary and backup paths optimal design for traffic engineering in hybrid IGP/MPLS networks.* Proceedings of the 7th International Workshop on the Design of Reliable Communication Networks (2009).

[13] D. Cvetkovic, M. Doob and H. Sachs. *Spectra of Graphs.* Academic Press, New York, 1979.

[14] G. Del Corso, A. Gullí and F. Romani. *Fast PageRank computation via a sparse linear system.* Internet Math., 3-2 (2005), pp. 259–281.

[15] M. Donatelli. *A note on grid transfer operators for multigrid methods.* Manuscript 2008, arXiv:0807.2565v1.

[16] A. Frangioni, G. Gallo *A bundle type dual-ascent approach to linear multicommodity Min Cost Flow problems.* INFORMS J. Comput. 11 (1999), pp. 370–393.

[17] A. Frangioni, B. Gendron *0-1 reformulations of the multicommodity capacitated network design problem.* Disc. Appl. Math., 157 (2009), pp. 1229–1241.

[18] A. Frangioni, C. Gentile. *New Preconditioners for KKT Systems of Network Flow Problems.* Siam J. Opt., 14 (2004), pp. 894–913.

[19] A. Frangioni, C. Gentile. *Prim-based BCT preconditioners for Min-Cost Flow Problems.* Computational Optimization and Applications, 36 (2007), pp. 271–287.

[20] A. Frangioni, S. Serra Capizzano. *Spectral analysis of (sequences of) graph matrices.* Siam J. Matrix Anal. Appl., 23-2 (2001), pp. 339-348.

[21] G.H. Golub, C. F. Van Loan. *Matrix computations.* North Oxford Academic, 1983.

[22] R. Horn, S. Serra-Capizzano. *A general setting for the parametric Google matrix.* Internet Math., 3-4 (2008) pp. 385–411.

[23] H.B. Keller. *Numerical methods for two-points boundary-value problems.* Blaisdell, London, 1968.

[24] G. Kirchhoff. *Uber die Auflosung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Strome gefuhrt wird.* Ann. Phys. Chem. 72 (1847) 497–508. Translated by J. B. O'Toole in I.R.E. Trans. Circuit Theory, CT-5 (1958) 4.

[25] A. Langville, C. Meyer. *A survey of eigenvector methods for WEB information retrieval.* Siam Review, 47-1 (2005) pp. 135–161.

[26] B. Mohar. *Some Applications of Laplace Eigenvalues of Graphs.* Graph Symmetry: Algebraic Methods and Applications, Eds. G. Hahn and G. Sabidussi, NATO ASI Ser. C 497, Kluwer, 1997, pp. 225–275.

[27] R.D.C. Monteiro, J.W. O'Neal and T. Tsuchiya. *Uniform boundedness of a preconditioned normal matrix used in interior-point methods.* Siam J.Opt, 15-1 (2004), pp. 96–100.

[28] M. Ng, S. Serra-Capizzano and C. Tablino Possio. *Multigrid preconditioners for symmetric Sinc systems.* ANZIAM J., 45-E (2004), pp. 857–869.

[29] D. Noutsos, S. Serra-Capizzano and P. Vassalos. *The conditioning of FD matrix sequences coming from semi-elliptic Differential Equations.* Linear Algebra Appl., 428-2/3 (2008), pp. 600–624.

[30] R. Olfati-Saber, R.M. Murray. *Consensus problems in networks of agents with switching topology and time-dealays.* IEEE Trans. Automatic Control, 49-9 (2004), pp. 1520–1533.

[31] L.F. Portugal, M.G.C. Resende, G. Veiga and J.J. Jùdice. *A truncated primal-infeasible dual-feasible network interior point method* Networks, 35 (2000), pp. 91–108.

[32] J.W. Ruge, K. Stüben. *Algebraic multigrid.* In Multigrid methods, vol. 3 of Frontiers Appl. Math., pp. 73–130. Siam, Philadelphia, PA, 1987.

[33] Y. Saad. *Iterative Methods for Sparse Linear Systems.* PWS, Boston, 1996.

[34] J. Schwartz, A. Steger and A. Weissl. *Fast Algorithms for Weighted Bipartite Matching.* in *Experimental and Efficient Algorithms*, Lecture Notes in Computer Science 3503, pp. 476–487, 2005.

[35] M. Semplice, M. Donatelli and S. Serra-Capizzano. *Preconditioned implicit PDE solvers for degenerate parabolic equations with applications to monument conservation.* Submitted.

[36] S. Serra Capizzano. *Multi-iterative methods.* Comput. Math. Appl., 26-4 (1993), pp. 65–87.

[37] S. Serra-Capizzano, C. Tablino Possio. *Multigrid methods for multilevel circulant matrices.* Siam J. Sci. Comput., 26-1 (2004), pp. 55–85.

[38] H.D. Sterck, T.A. Manteuffel, S.F. Mccormick, Q.Nguyen and J. Ruge. *Multilevel Adaptive Aggregation for Markov Chains, with Application to WEB Ranking.* Siam Journal on Scientific Computing, 2007.

[39] K. Stüben. *A review of algebraic multigrid.* J. Comput. Appl. Math., 128 (2001) 281-309.

[40] U. Trottenberg, C. Oosterlee and A. Schuller. *Multigrid.* Academic Press, 2001.