Computational evaluation of multi-iterative approaches for solving graph-structured large linear systems

Pietro Dell'Acqua [*] Antonio Frangioni[†]

Stefano Serra-Capizzano[‡]

June 28, 2014

Abstract

We analyze the practical efficiency of multi-iterative techniques for the numerical solution of graph-structured large linear systems. In particular we evaluate the effectiveness of several combinations of coarser-grid operators which preserve the graph structure of the projected matrix at the inner levels and smoothers. We also discuss and evaluate some possible strategies (inverse projection and dense projection) to connect coarser-grid operators and graph-based preconditioners. Our results show that an appropriate choice of adaptive projectors and tree-based preconditioned conjugate gradient methods result in highly effective and robust approaches, that are capable to efficiently solve large-scale, difficult systems, for which the known iterative solvers alone can be rather slow.

Keyword graph matrices, multigrid, conditioning and preconditioning.

1 Introduction

Large linear systems with (weighted) graph-structured matrices can be found in many applications (e.g. [13] and references therein); among others, we may refer to problems in Web searching engines [28], general Markov chains [41], consensus algorithms [35] and optimization problems in networks [1, 5]. In this note, the specific application motivating our research is the last, where the involved structures are inherently symmetric and positive definite since the matrices arise from (weighted) graph *Laplacian* operators [13, 29]. More precisely, we are interested in the efficient solution of linear systems occurring at all iterations of Interior Point (IP) techniques for the Min-Cost Flow (MCF) problem, which can be briefly described as follows. Given a connected directed graph $\mathcal{G} = (\mathcal{U}, \mathcal{V})$,

[*]Dipartimento di Ingegneria Navale, Elettrica, Elettronica e delle Telecomunicazioni, Università di Genova, Via all'Opera Pia 11, 16145 Genova, Italy. E-mail: pietro.dellacqua@gmail.com

[†]Dipartimento di Informatica, Università di Pisa, Largo B. Pontecorvo 3, 56127 Pisa, Italy. E-mail: frangio@di.unipi.it

[‡]Dipartimento di Scienza e alta Tecnologia, Università dell'Insubria - sede di Como, via Valleggio 11, 22100 Como, Italy. E-mail: stefano.serrac@uninsubria.it

where $\mathcal{U}$ is the set of nodes and $\mathcal{V}$ is the set of arcs, with $|\mathcal{U}| = n$ and $|\mathcal{V}| = m$, the MCF problem is formulated as the Linear Program (LP)

$$\min \, \{ \, cx \; : \; Ex = d \; , \; 0 \leq x \leq u \, \},$$

where E is the node-arc incidence matrix of $\mathcal{G}$ (the $n \times m$ matrix such that $E_{ia} = 1$ if arc a emanates from node i, $E_{ia} = -1$ if a terminates at i, and $E_{ia} = 0$ otherwise), $c = [\, c_a \,]_{a \in \mathcal{V}}$ is the vector of arc costs, $u = [\, u_a \,]_{a \in \mathcal{V}}$ is the vector of arc upper capacities, $d = [\, d_i \,]_{i \in \mathcal{U}}$ is the vector of node deficits, and $x = [\, x_a \,]_{a \in \mathcal{V}}$ is the vector of flows. The *flow conservation constraints* $Ex = d$ model the flow traveling in the graph from *sources* (nodes with $d_i > 0$) to *destinations* (nodes with $d_i < 0$), possibly passing through the remaining *transshipment nodes* (with $d_i = 0$). IP methods, which have grown a well-established reputation as efficient algorithms for large-scale linear and nonlinear problems, can be successfully applied to the MCF problem [36, 30, 19, 20], possibly in combination with combinatorial algorithms [21]. For this to happen, however, one has to solve at each step linear systems of the form

$$E \Theta E^T x = b, \tag{1}$$

where $b \in \mathbb{R}^n$ and the $m \times m$ diagonal matrix Θ, with positive diagonal entries, depends on the IP iteration. Clearly, problem (1) inherits a strong structure of (weighted) graph, where the diagonal entries of Θ specify the arc weights. Indeed $L = E \Theta E^T$ is the well-known *Laplacian* of the weighted undirected graph obtained from $\mathcal{G}$ by ignoring the orientation of the arcs and by summing up the weights of all arcs, which "collapse" in the same edge: we refer the reader to [11, 27, 29] for many applications of the Laplacian of a graph in such diverse fields as graph theory, statistics, and combinatorial optimization. While in general-purpose IP-based LP solvers, the linear system in (1) is typically solved by means of direct methods, the same numerical strategy cannot be applied to very large sparse networks as disastrous fill-in may occur [10], which leads to an unbearable iteration cost. Thus we have to apply iterative methods, which motivates the study of the spectral behavior of L [22, 30], since the convergence speed of the considered iterative solvers strongly depends on the spectrum. The conditioning of L clearly depends on both $\mathcal{G}$ (fixed) and Θ (changing with the IP iterations). While in the first IP iterations, the matrix Θ is close to the identity and the spectral difficulties are mild [22], in the last IP iterations Θ becomes highly ill-conditioned, and so does L. Thus, especially at the last IP iterations some device, such as *preconditioning*, is required in order to make the considered iterative techniques efficient.

A simple yet successful approach is to solve (1) through a preconditioned conjugate gradient (PCG) method using *subgraph preconditioners* [7, 8] of the form

$$L_{\mathcal{S}} = E_{\mathcal{S}} \Theta_{\mathcal{S}} E_{\mathcal{S}}^T \tag{2}$$

where $E_{\mathcal{S}}$ and $\Theta_{\mathcal{S}}$ denote the restriction of E and Θ, respectively, on the arcs of a "simple" subgraph $\mathcal{S} = (\mathcal{U}, \bar{\mathcal{V}})$ of $\mathcal{G}$ ($\bar{\mathcal{V}} \subseteq \mathcal{V}$): the latter choice can also be improved, in appropriate circumstances, by the following modification

$$L'_{\mathcal{S}} = L_{\mathcal{S}} + \rho \cdot \mathrm{diag}(\, L - L_{\mathcal{S}} \,) \tag{3}$$

where $\mathrm{diag}(A)$ is the diagonal matrix, having as diagonal elements those of A, and $\rho > 0$ is an appropriately chosen weight. The most effective choices for $\mathcal{S}$

are chordal-type graphs [36, 30, 19, 20], which avoid fill-in in the preconditioner and therefore allow to keep low the related iteration cost. Furthermore, it is well-known that the matrices E and L have not full rank. In particular, $e^T E = 0^T$ where e is the all-ones vector of appropriate length. However, since $e^T d = 0$ as well, the linear system (1) has infinitely many solutions. One could therefore solve (1) by considering a special invertible reduced system. However, for positive semidefinite matrices, CG or PCG methods, with zero initial vector, are well-known to provide a solution in the least-squares sense. Hence, the PCG technique allows to work on the original graph, which is not true for other iterative approaches.

The PCG iteration using subgraph preconditioners often works quite well, but there are some cases where the convergence rate is slow. The objective of this paper is therefore not to replace the subgraph-based PCG approach, but rather to complement it with ideas from the algebraic multigrid field [37, 42, 9]—and in particular from multi-iterative techniques [39]—to further improve the efficiency, in particular in the "difficult" cases. Let us remark that, while our main application is the MCF problem, graph-structured linear systems like (1) are very common. Indeed subgraph preconditioners have been studied and found effective in many other applications as well [7, 8] and hence our study may have a wider applicability. We refer in particular to the recent results of [16], where we characterized the properties required by coarser-grid operators in a multigrid setting to preserve the graph structure of the projected matrix at the inner levels. Motivated by these results, we evaluate the effectiveness of several different combinations between coarser-grid operators and smoothers. We also discuss and evaluate some possible strategies (inverse projection and dense projection), which highlight connections between coarser-grid operators and preconditioners. Our findings show that an appropriate choice of adaptive projectors and tree-based PCG methods result in highly effective and robust algorithms, that are capable to efficiently solve large-scale, difficult systems for which the known techniques are rather slow.

The paper is organized as follows. In Section 2 and Section 3 we rapidly recall the multigrid procedure and the theoretical results related to structure-preserving projectors. In Section 4 we discuss few families of projectors which provide indications on how effective projections have to be chosen. In Section 5 we propose two possible ideas, inverse projection and dense projection, to simultaneously choosing the projector and the graph preconditioner. Finally Section 6 is devoted to a critical discussion of the numerical experiments and in Section 7 conclusions are drawn.

2 Multigrid and structure-preserving projectors

When considering a linear system as in (1), with a symmetric positive definite $n \times n$ matrix $L = E\Theta E^T$, the standard *V-cycle method* [43] proceeds by choosing a *number of levels* $l \in (0, n)$ (most often $l \approx \log n$). Each level has a fixed size $n_0 = n > n_1 > n_2 > \ldots > n_l > 0$, corresponding to the size of the projected system that is obtained from the previous level by means of the $n_{i+1} \times n_i$ full-rank *projector* R^i_{i+1}. Also two classes $\mathcal{P}_i$ and $\mathcal{Q}_i$ of iterative methods for n_i-dimensional linear systems are selected, most often among standard stationary iterative methods [44], such as Richardson, (damped) Jacobi, Gauss-Seidel etc, with prescribed iteration matrix: they are called *smoothers*. More in detail, the global multigrid iteration is based on (recursively) first applying some steps of the *pre-smoother* $\mathcal{P}_i$, then projecting the system on the lower level through R^i_{i+1}, performing sub-grid correction and interpolation, and then improving the result with a few iterations of *post-smoother* $\mathcal{Q}_i$. Thus, the algorithm has essentially two degrees of indetermination: the choice of the projectors R^i_{i+1} and the choice of the smoothers $\mathcal{P}_i$, $\mathcal{Q}_i$. In the following we will only consider convergent smoother iterations, in order to insure that the multigrid iteration matrix has L-norm smaller (in other words, is never worse) than that of the smoother alone [39, 24]. In the remainder of this section, we briefly discuss the choice of smoothers and projectors, from the viewpoint of the computational performances.

2.1 Smoothers

We started our experiments (see [16]) by testing several combinations of "simple" pre- and post-smoothers, such as Gauss-Seidel with relaxation parameters between 1/2 and 1, CG and PCG with elementary preconditioners, to find the one with better spectral complementarity [39] and therefore performances. The preliminary numerical results were not encouraging, especially in the last iterations of the IP process, where the entries of Θ become very unbalanced, with very large ($\approx$ `1e+6`) and very small ($\approx$ `1e-10`) ones. A similar occurrence, with a wild behavior of the diagonal entries, was already experienced in a different context [31]; in that application, the only successful smoother was a PCG method with a carefully chosen "powerful" preconditioner. This motivated our choice to restrict our subsequent tests to PCG with specialized preconditioners such as diagonal, incomplete and strongly incomplete Cholesky, and subgraph-based ones.

Besides the choice of the iterative methods, one important decision is also the *number of steps* ($\nu_{i,\mathrm{pre}}$ and $\nu_{i,\mathrm{post}}$ for the pre- and post-smoother, respectively, at level i) to be performed. If the cost at every level of the MG iteration is linear with respect to the dimension, the overall arithmetic cost W is still linear even if the number of steps grows as i^k with the level i. Indeed one can easily prove (see e.g. [16]) that

$$W = O\big(\, n \textstyle\sum_{i=1}^{\infty} i^k / 2^{i-1} \,\big) \qquad (4)$$

where the series in (4) is convergent, with initial values for $k = 0, 1, 2$ being respectively 2, 4 and 12. 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. In general the increased computational cost per multigrid iteration is often not sufficiently compensated by the decrease in the number of iterations. Therefore in all the subsequent test we elected to use the simplest setting $k = 0$, i.e., one single pre-smoothing iteration and one single post-smoothing iteration.

2.2 Projectors

Our theoretical and initial results [16] quickly made clear that standard operators with good performances in the context of differential equations, such as the very classical Full Weighting Operator (FWO, see [43]), were not performing efficiently in the context of general graph matrices. The rationale behind this behavior was found to be that the FWO is a *graph operator* for the very special graph matrix of Poisson problems, which is basically the matrix of a *linear graph.* A graph operator (cf. Section 3) is defined as a projector that preserves the graph structure in the lower level. Similar notions have been developed with good results in different contexts, for example when designing multigrid solvers for Markov Chains [41]. In practice one can restrict to Aggregation Operators, so that the matrix in the lower level is the Laplacian of a graph corresponding to the *aggregation of nodes* of the original graph. For problem of our interest, the linear interpolation, associated with FWO, is not effective, even for "easy" graphs and moderate conditioning [22]. This is also confirmed by the fact that using more sophisticated choices, such as quadratic interpolation [17] and even cubic interpolation [43] does not lead to better results (actually, even to worse ones). Thus the differential setting is of no help for general graphs and arc weights (such as these of MCF matrices at final IP iterations).

To conclude, the preliminary experiments showed that the only effective smoothers are sophisticated preconditioned Krylov methods and the only effective projectors are structure-preserving ones. In the next section we rapidly recall the theoretical results of [16] about graph operators, paving the way for our subsequent computational results.

3 Graph operators

We are interested in projection operators that preserve the graph structure of the matrix. We call an operator R a *graph* operator if, given any incidence matrix E, $RE = E'\Theta'$ where E' is an incidence matrix and Θ' is an invertible diagonal matrix. In (1), one then has

$$RE\Theta E^T R^T \quad = \quad E'\Theta'\Theta\Theta' E'^T \quad = \quad E'\tilde{\Theta} E'$$

where $\tilde{\Theta} = \Theta'\Theta\Theta'$ is a diagonal matrix with positive diagonal entries by the assumption concerning Θ in (1). Hence a graph operator preserves the graph structure at the lower levels, in the sense that the projected problem is still a (smaller) weighted Laplacian matrix. A weaker notion is that of *graph operator for a given matrix E*, which requires the property to hold for the specific E, although it may not hold for all possible incidence matrices.

Actually, the above property is not enough. Indeed, we need that operators R that be also *admissible*, i.e., do not have any column with all zero elements and any row with all equal elements. In fact, all-zero column means that a node is "ignored", so that the projection 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' has an isolated node; furthermore, since the MG conditions impose that R has to be of full rank, at most only a unique row could have all equal elements. Clearly, if R is admissible then $\mathrm{nz}(R) \geq n$, where $\mathrm{nz}(\cdot)$ denotes the number of non-zero elements. Of particular relevance are *minimum* operators, which are all admissible graph operators R such that $\mathrm{nz}(R) = n$.

A characterization of the set of minimum graph operators is provided in [16], along the following lines:

- R is a *constant column sums* (CCS) operator if the sum of elements of every column is constant;
- R is a *zero column sum* (ZCS) operator if it conserves the property of E that all columns have zero sum;
- R is a ZCS operator if and only if R is a CCS operator;
- any minimum operator is a binary matrix up to a scalar factor;
- for any admissible graph operator R, $R = M + C$ where C is a row matrix (each of its rows is composed by all identical numbers) and M is a minimum operator;
- for any admissible graph operator R that is also a binary matrix, then either $R = M$ or $R = U - M$ where M is a minimum operator and U is the all-ones matrix.

Actually the last two results only hold if R has more than two rows, but this is clearly not an issue in practice. What these results say is that, basically, any admissible graph operator that is a binary matrix is either a minimum operator or the "complement" of a minimum operator. In turn all minimum operators are binary matrices up to a scalar factor, and therefore restricting our attention to Aggregation Operators seems to be a reasonable choice. In addition, by the above reasoning, the action on an incidence matrix E of any graph operator R and of the related minimum operator M is the same.

4 Minimum operators

As discussed in Section 3, the set of non-minimum operators, as a subset of admissible graph operators, is not empty, although it just contains row matrix "perturbations" of minimum operators. One may therefore wonder whether non-minimum operators could be preferable to minimum ones. In practice, we did not identify any promising way to construct non-minimum operators. This does not mean that further research may not identify effective ways to do that, maybe if the graph has a particular structure; however, it justifies the fact that in the following, we will concentrate on multigrid methods 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 connecting nodes combined together just disappear. Obviously a fundamental question has now to be answered, i.e., how to select the subset of nodes to be combined together.

There are many possible ways to approach this question. For instance, if $\mathcal{G}$ is a tree, then (1) can be solved within $O(n)$ arithmeric operations [1]. More in general, if $\mathcal{G}$ is a chordal-type graph, then (1) can be solved by using $O(m)$ arithmetic operations [19, 20]: in fact, the idea of subgraph-based preconditioners is based on this computational result. A promising idea may be that of choosing the aggregation in such a way that after a few levels the resulting graph is chordal, 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 MG iterations. In the following we will describe some classes of minimum operators that we have devised and tested to start shedding 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 Oblivious projectors

We now present a first list of simple aggregation projectors. We call them *oblivious* operators, as their form is independent of the matrix to which they are applied. Oblivious operators are attractive from the computational viewpoint because they can be computed a-priori, therefore they are as cheap as possible. For instance, the First Operator aggregates the first $d+1$ nodes into one:

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

In a similar way, the Last Operator aggregates the last $d+1$ nodes into one, the Extreme Operator aggregates the first and the last $(d+1)/2$ nodes into one, the Medium Operator aggregates the $d+1$ "central" nodes into one, the Couple Operator aggregates every odd-indexed node with the following one, in other words it is the product of $\approx n/2$ pair operators corresponding to pairs $(1,2)$, $(3,4)$, ... Finally, 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). Clearly, it is possible to alternate two or more of these operators along MGM levels.

Although these are the projectors used with success in preliminary tests of Subsection 2.2, we found that the effectiveness of these operators is strongly influenced by the structure of the underlying graph $\mathcal{G}$. Thus, it is difficult to choose any fixed oblivious 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. In other words, we have to consider non-oblivious operators, despite the fact that they can more expensive to determine.

4.2 Adaptive projectors

Several *adaptive* operators can be devised, depending on the values of the matrix at hand. Since they may be more expensive from the computational viewpoint, a non-trivial trade-off, between increasing the convergence speed and the associated computational cost, has to be found. Here we briefly recall a set of operators which are very simple to create, compared to other sophisticated techniques (such as iterative pairwise minimal operators, strength-based aggregation operators, strength-based algebraic multigrid operators, and combinatorial operators [16]), and nonetheless show competitive performances. The idea is to construct adaptive operators by using 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. Clearly different operators arise as a consequence of different rules for choosing 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 1 *A graph operator R is a* contraction operator *for E if it aggregates exclusively adjacent nodes in $\mathcal{G}$.*

Definition 2 *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.*

Contraction operators are attractive, since they do not *damage* the graph structure. Moreover this is useful when we employ subgraph-based preconditioners. Because of this we have mostly concentrated on these while developing adaptive operators. In particular, an 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 nine "$\{x, y\}$" 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}. $\{x, \max\}$ projectors substantially outperform the corresponding variants where j is selected either at random or by looking at arcs with small weight. It is easy to see that for the class of $\{x, \max\}$ operators, contraction property holds.

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 (for similar conclusion in a different perspective see e.g. [14, 32, 33] and references therein). These results seem to indicate that there could be still room for improvements, although they also very clearly indicate that already these simple adaptive projectors are much better than oblivious ones. This is shown in Table 1 on a subset of instances for one of the most promising variant of the MG method (see Section 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, convergence data are reported regarding the first two iterations (where $\Theta \approx I$), the last iteration k and the second last $k-1$ (where Θ is highly unbalanced) and some middle ones.

MGM$_7$	`NET`		`GRID`		`GOTO`	
IP iteration	OBL	ADA	OBL	ADA	OBL	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 1: Comparison of oblivious and adaptive projectors in terms of number of iterations

As Table 1 shows, whereas the best oblivious approach (OBL) is competitive with an adaptive one (ADA) in some “easy” cases, the latter is the only option to achieve uniformly acceptable performances.

It is clear that different and possibly more sophisticated choices of R may exist, even by restricting the focus 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

$$L[i,j] = \begin{bmatrix} L_{ii} & L_{ij} \\ L_{ji} & L_{jj} \end{bmatrix}$$

where $L_{ii} > 0$, $L_{jj} > 0$, $L_{ij} = L_{ji} < 0$ and $\det(L[i,j])$ is positive. Because weak dominance for rows holds, that is $\sum_{j \neq i} |L_{ij}| \leq L_{ii}$ with strict inequality at least for one index (if R is a graph operator, then this is true at every MGM level), it follows 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 one 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 measure how well or badly conditioned the 2×2 minor is, in order to estimate how promising a (i,j) pair is. This leads to max-minor or min-minor operators, depending on whether one chooses to preferentially aggregate well-conditioned or ill-conditioned minors. Computational experiments show 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.

5 Relationships with preconditioning

As anticipated in Subsection 2.1 and numerically illustrated in the next one, the PCG with subgraph-based preconditioners is the most promising class of methods to serve as effective pre- and post-smoothers in a MG context. Thus, at least by restricting to *contraction operators* R, one actually has 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. In order to do this, 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 that S is positive definite and in fact "easy" to invert, which requires specific care in the choice of $\mathcal{S}$ [19, 20].

5.1 Inverse projection

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

$$P_k = (R_kR_k^T)^{-1}R_kS^{-1}R_k^T(R_kR_k^T)^{-1}R_kLR_k^T \tag{5}$$

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 1 *For all $A, B \in \mathbb{C}^{n\times n}$ the characteristic polynomials of AB and BA coincide and therefore $\sigma(AB) = \sigma(BA)$, with $\sigma(X)$ denoting the spectrum of a square matrix X.*

Lemma 2 *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 1 *With the notations given at the beginning of Subsection 5.1, 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 define $Q_k = R_k^T(R_kR_k^T)^{-1}R_k$ eith R_k given in (5). Thanks to Lemma 2, 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 2, 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 (see [6]) to estimate the first non-zero eigenvalue ($\lambda_{n-n_k+1} = \beta_1$):

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

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

$$\begin{aligned}
\lambda_{n-n_k+1} &\geq \min_{\dim(V)=n-n_k+1} \max_{z\in Z_V} \frac{z^T Q_k L Q_k z}{z^T S z} \\
&= \min_{\dim(V)=n-n_k+1} \max_{z\in Z_V} \frac{z^T L z}{z^T S z} \geq \alpha_1 \geq a
\end{aligned}$$

and the proof is concluded. •

In the case where $[a_k, b_k] \subset [a, b]$, 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 we have good hopes that the same holds for the practical behavior of the PCG. Theorem 1 instead only gives indications about the lower extreme of the interval. However the lower estimate is the most important one for PCG convergence, as well emphasized in [3], where it is proven that eigenvalues close to zero deteriorate the convergence speed much more than large eigenvalues. Therefore a good practical behavior of the PCG smoother might be expected at all levels of the MG method, 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

$$S_k = R_k S R_k^T$$

as preconditioner at the k-th level. Here 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 $\mathcal{G}$: the clear advantage is that the computation of S_k is very inexpensive. However, the number of non-zero elements can significantly increase with respect to that of the original subgraph $\mathcal{S}$ (chordal subgraphs are typically fairly sparse), thus causing a relevant growth in the cost for inverting S_k. Yet, it is possible to choose $\mathcal{R}$ appropriately in order to avoid this issue.

Theorem 2 *Let $\mathcal{S}$ be a chordal graph, $(i, j) \in \mathcal{S}$ and $\mathcal{S}'$ obtained by $\mathcal{S}$ by aggregating i and j. Then $\mathcal{S}'$ is a chordal graph.*

Proof Basically the proof is based on the fact that aggregating an existing arc cannot create any new cycle. Indeed, 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 1, 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.

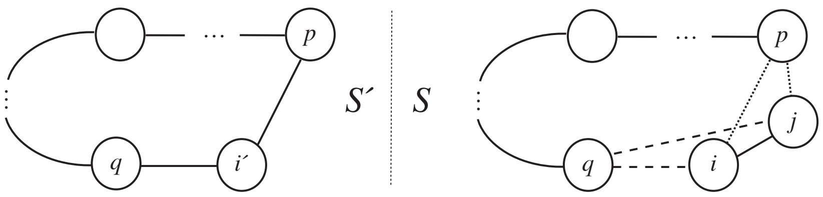

Figure 1: Aggregation in a chordal graph preserves the property

This concludes the proof. •

Thus, choosing $\mathcal{R} \subset \mathcal{S}$, the triangular structure is preserved and we avoid 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*, one of the most effective more-than-tree subgraph-based preconditioners [19, 20]. Roughly speaking, these structures are obtained by adding this kind of arcs to an existing (or being constructed) spanning tree. In other words, if $\mathcal{S}$ is a tree and $(i,j) \notin \mathcal{S}$ is an arc joining brothers, which may be deemed useful to be part of $\mathcal{R}$, then (i,j) has to be added to $\mathcal{S}$ in the first place. It appears that more-than-tree chordal preconditioners may be even more attractive in the context of MG 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. Consequently, 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, and it is more delicate in the MGM setting. 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 appropriate for finding the right balance between the selection of the preconditioner and that of the projector. Choosing $\mathcal{R}$ and $\mathcal{S}$, in such a way that good convergence results at all levels are combined 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 the MG method on instances of equation (1) coming from real applications, in particular when considering MCF problems. Having discussed the issue of the choice of the projector R in the previous sections, here we concentrate on the other fundamental ingredient of a successful MG method, i.e., the appropriate selection of pre- and post-smoothers.

For our experiments, we compare several variants of MGM with a few reference Krylov methods: 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 the MG method are reported in Table 2. For each version we describe which smoother is chosen in the first, in the last and in the middle MG levels; $\mathbf{MGM_{7b}}$ alternates $\mathbf{PCG_3}$ with $\mathbf{PCG_2}$ along the levels. "Inverse projection" and "dense projection" refer to the PCG methods where the preconditioning schemes are those described in Section 5.

Method	pre-smoother			post-smoother		
	first	middle	last	first	middle	last
$\mathbf{MGM_0}$		Gauss-Seidel			CG	
$\mathbf{MGM_1}$		$\mathbf{PCG_1}$			$\mathbf{PCG_1}$	
$\mathbf{MGM_2}$	$\mathbf{PCG_3}$	$\mathbf{PCG_1}$		$\mathbf{PCG_3}$	$\mathbf{PCG_1}$	
$\mathbf{MGM_3}$	$\mathbf{PCG_3}$	$\mathbf{PCG_2}$	$\mathbf{PCG_3}$	$\mathbf{PCG_3}$	$\mathbf{PCG_2}$	$\mathbf{PCG_3}$
$\mathbf{MGM_{3b}}$	$\mathbf{PCG_3}$	$\mathbf{PCG_{2b}}$	$\mathbf{PCG_3}$	$\mathbf{PCG_3}$	$\mathbf{PCG_{2b}}$	$\mathbf{PCG_3}$
$\mathbf{MGM_4}$		$\mathbf{PCG_2}$			$\mathbf{PCG_2}$	
$\mathbf{MGM_{4b}}$		$\mathbf{PCG_{2b}}$			$\mathbf{PCG_{2b}}$	
$\mathbf{MGM_5}$	$\mathbf{PCG_3}$	inverse projection		$\mathbf{PCG_3}$	inverse projection	
$\mathbf{MGM_6}$	$\mathbf{PCG_3}$	dense projection		$\mathbf{PCG_3}$	dense projection	
$\mathbf{MGM_7}$		$\mathbf{PCG_3}$			$\mathbf{PCG_3}$	
$\mathbf{MGM_{7b}}$		$\mathbf{PCG_{3,2}}$			$\mathbf{PCG_{3,2}}$	

Table 2: List of MGM

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`. These have been used in several cases to produce (both single and multicommodity) flow test instances [10, 19, 20, 21]. Each generator produces matrices with different topological properties, as shown in Figures 2, 3, and 4. Furthermore the solution of the MCF instances via an IP methods produces weight matrices Θ with a different behavior.

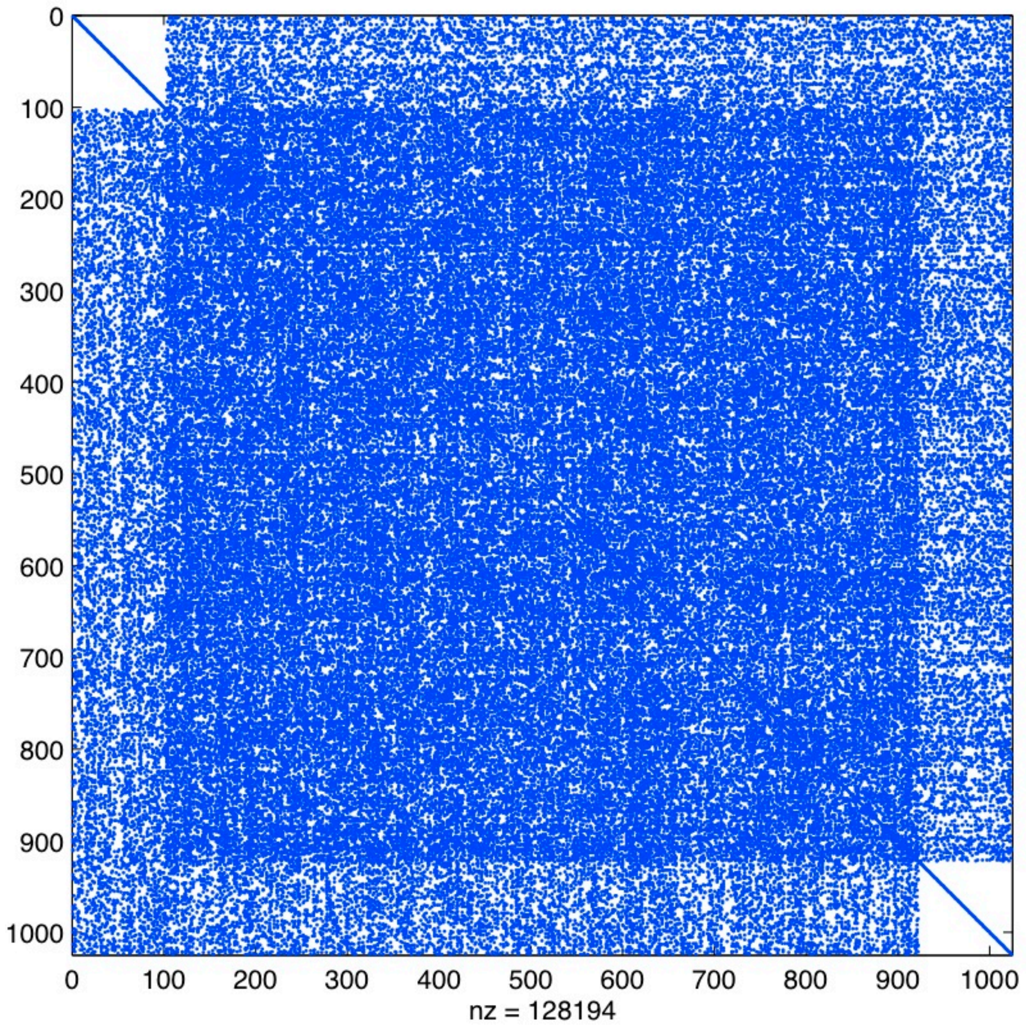

Figure 2: Structure of L for the Net problem

As the pictures of Figure 2 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 is considerably more difficult to solve than the former, both in terms of IP algorithm and of the corresponding linear systems. The difficulty of `Goto` is likely to be related to the structure of the L matrix, which is far from the block and the banded cases. 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` problems.

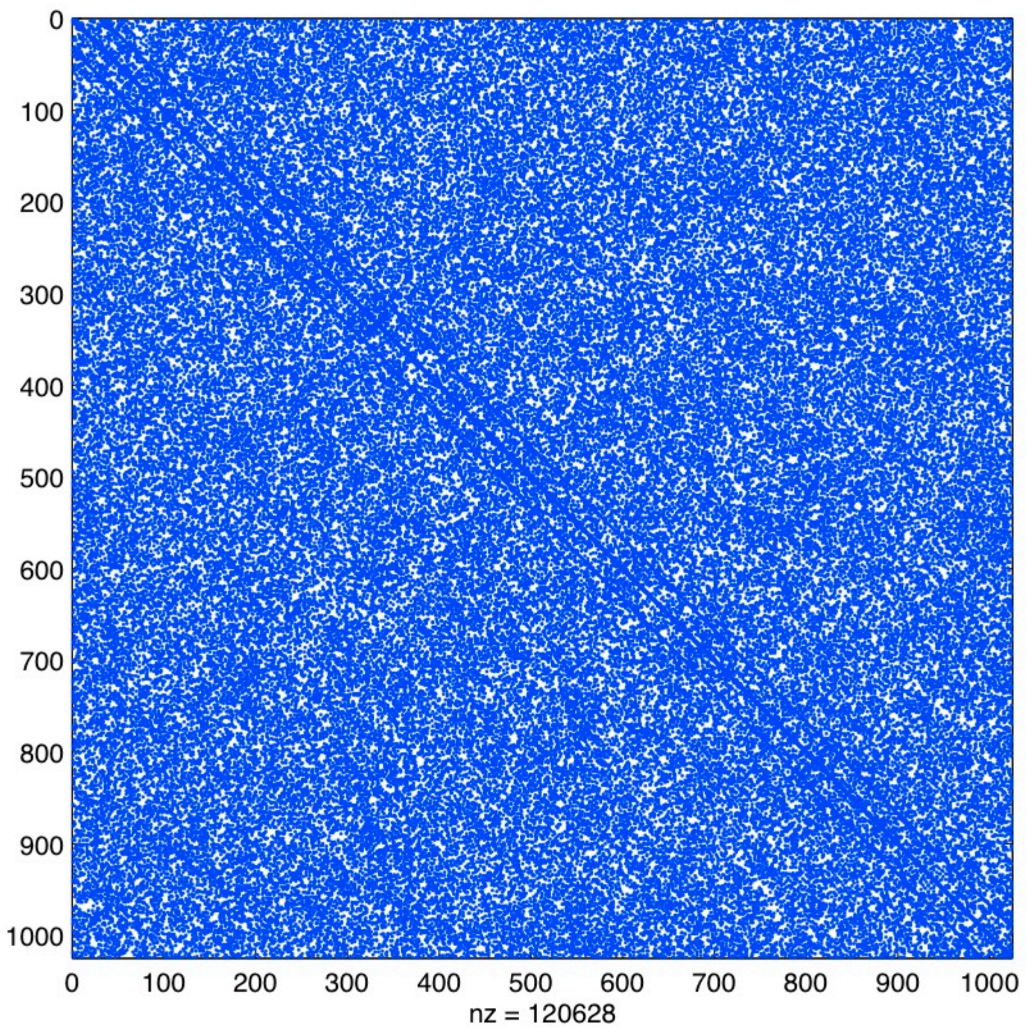

Figure 3: Structure of L for the Grid problem

6.2 Results

All methods showed convergence, which however in several cases was exceedingly slow. Thus, we had to resort in 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 the MGMs, considering that the latter have a higher cost per iteration. Tables 3, 4, and 5 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 reports results for the same matrix at different IP iterations, comprising the first iterations, middle iterations, and last iterations (k denotes the last IP iteration, cf. Subsection 4.2). Methods requiring more than the maximum allotted number—100 for multigrid methods, 5000 for conjugate gradient methods—of iterations are denoted by $*$ and $-$ respectively.

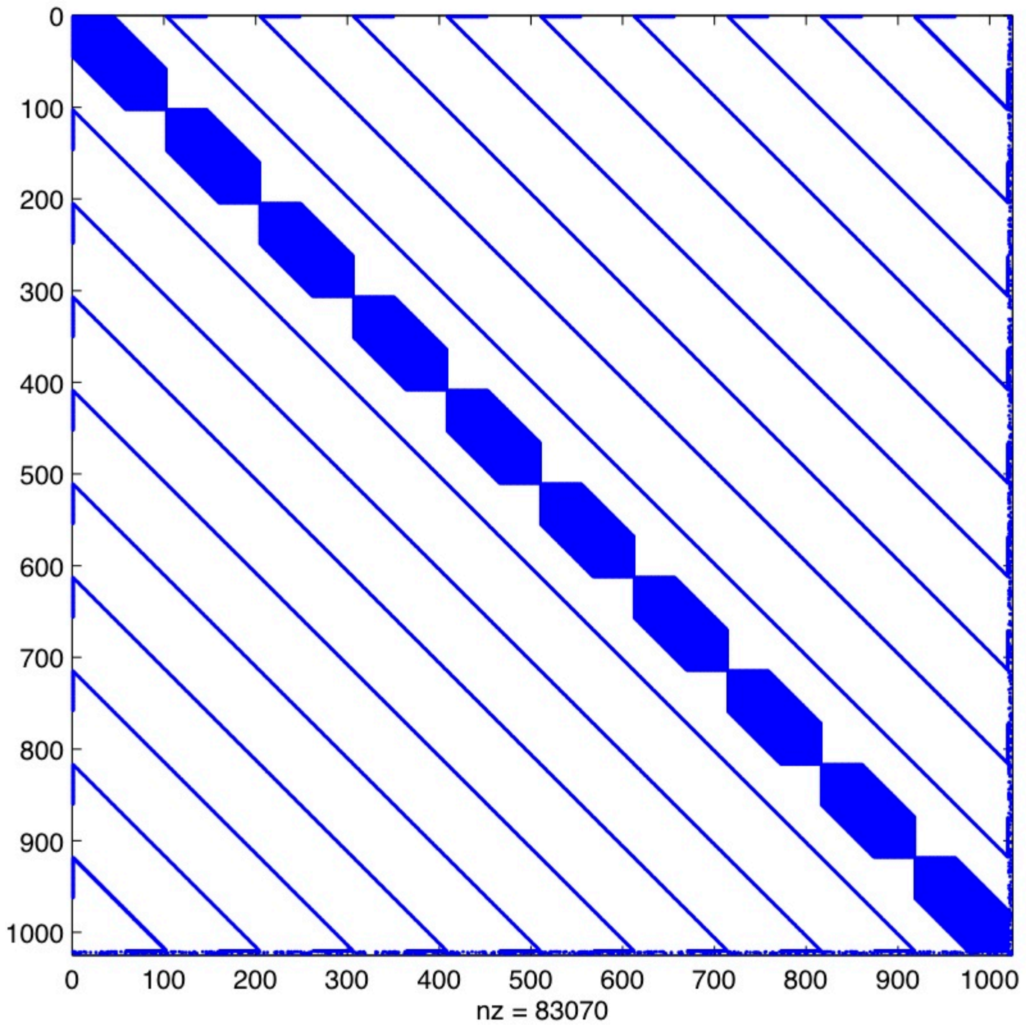

Figure 4: Structure of L for the Goto problem

The results confirm that subgraph-based preconditioners are by far the most effective Krylov technique (among the tested ones), in particular in the "difficult" last IP iterations where Θ is highly unbalanced. 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 MGMs (especially in the difficult cases) are those which make use—all or in part—of tree preconditioners, such as $\mathbf{MGM_3}$, $\mathbf{MGM_{3b}}$, $\mathbf{MGM_7}$ and $\mathbf{MGM_{7b}}$. Again Cholesky-based PCG smoothing can be competitive only in the initial IP iterations, but not in the other IP iterations, especially in the "difficult" `Goto` instances.

The case $\mathbf{MGM_0}$ is somewhat different from the remaining MGMs, since one iteration of this method is less expensive than one iteration of a sophisticated PCG. Therefore, in the first IP iterations of all problems $\mathbf{MGM_0}$ is

Method	1	2	$k/3$	$k/2$	$2k/3$	$k-1$	k
CG	41	51	63	1433	$*$	$*$	$*$
$\mathbf{PCG_1}$	8	16	28	140	1213	4789	2431
$\mathbf{PCG_2}$	8	7	8	17	109	473	221
$\mathbf{PCG_{2b}}$	5	7	7	35	282	1172	652
$\mathbf{PCG_3}$	8	10	11	21	14	6	3
$\mathbf{MGM_0}$	8	9	13	–	–	–	–
$\mathbf{MGM_1}$	5	10	13	–	–	–	–
$\mathbf{MGM_2}$	5	6	6	9	8	2	2
$\mathbf{MGM_3}$	5	6	7	11	7	2	2
$\mathbf{MGM_{3b}}$	5	6	7	11	8	2	2
$\mathbf{MGM_4}$	5	4	4	8	63	–	–
$\mathbf{MGM_{4b}}$	5	4	4	8	–	–	–
$\mathbf{MGM_5}$	5	6	6	11	10	2	2
$\mathbf{MGM_6}$	5	6	6	11	6	2	2
$\mathbf{MGM_7}$	5	6	7	11	6	2	2
$\mathbf{MGM_{7b}}$	5	6	7	11	7	2	2

Table 3: Results for `NET` instances

competitive compared with PCG techniques. However, this crucially depends on an appropriate choice of projector, as the performances of the simple MGM are strongly influenced by R. Conversely the MG methods with more powerful preconditioners are much more resilient to suboptimal choices of the projector.

An interesting performance measure to estimate the effectiveness of a MG method is

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

which is reported in Table 6. Since $\nu_{i,\mathrm{pre}} = \nu_{i,\mathrm{post}} = 1$ (we only perform one iteration of pre- and post-smoother), it is reasonable to compare performances by iteration steps, considering that the MGM cost will be at least twice—and likely around four times, cf. Subsection 2.1—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) θ_p gets larger than 4 and it can get above 40. Therefore the higher speed of convergence of MGM surely implies that the method is 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.

Method	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 4: Results for GRID instances

7 Conclusions

We have studied the practical efficiency of multi-iterative techniques for the numerical solution of graph-structured large linear systems. Motivated by some recent theoretical results characterizing the properties required from coarser-grid operators to preserve the graph structure of the projected matrix at the inner levels, we have evaluated the effectiveness of several different combinations of coarser-grid operators and smoothers. We have shown that an appropriate choice of adaptive projectors and tree-based preconditioned conjugate gradient methods, possibly involving (simple) schemes like *inverse projection* and *dense projection* to co-ordinate the choice of the preconditioner and that of the projector, result in highly effective and robust methods that are capable to efficiently solving large-scale, difficult systems, for which the known iterative techniques are rather slow.

An important challenge would be the extension of such a study to a non-symmetric setting (a typical example is given by the classic Google matrix [28, 15, 12]) and this will be a subject of future investigations.

References

[1] R.K. Ahuja, T.L. Magnanti, J.B. Orlin. *Network Flows: Theory, Algorithms and Applications.* Prentice Hall, Englewood Cliffs, NJ, 1993.

[2] A. Aricó, M. Donatelli, 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.

Method	1	2	$k/3$	$k/2$	$2k/3$	$k-1$	k
CG	27	917	$*$	$*$	$*$	$*$	$*$
PCG$_1$	17	270	1956	$*$	$*$	$*$	$*$
PCG$_2$	15	35	311	1676	$*$	$*$	$*$
PCG$_{2b}$	8	39	614	1359	2848	$*$	4443
PCG$_3$	16	55	183	342	455	405	385
MGM$_0$	17	90	–	–	–	–	–
MGM$_1$	10	70	–	–	–	–	–
MGM$_2$	8	15	47	54	73	66	75
MGM$_3$	6	11	44	15	10	9	9
MGM$_{3b}$	5	12	45	21	15	14	14
MGM$_4$	6	11	75	–	–	–	–
MGM$_{4b}$	6	13	66	–	–	–	–
MGM$_5$	10	23	–	–	–	–	–
MGM$_6$	6	15	48	98	–	–	–
MGM$_7$	6	12	39	22	16	9	9
MGM$_{7b}$	6	10	39	15	21	9	9

Table 5: Results for `GOTO` instances

θ_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 6: Relative efficiency of best PCG vs. best MGM

[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, 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, 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] M. Brezina, R.D. Falgout, S. MacLachlan, T.A. Manteuffel, S.F. McCormick, J. Ruge. *Adaptive algebraic multigrid.* SIAM J. Sci. Comput., 27 (2006), pp. 1261-1286.

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

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

[12] A. Cicone, S. Serra-Capizzano. *GOOGLE PageRanking problem: the model and the analysis*, J. Comput. Appl. Math., 234-11 (2010), pp. 3140–3169.

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

[14] H. De Sterck, T.A. Manteuffel, S.F. McCormick, K. Miller, J. Pearson, J. Ruge, G. Sanders. *Smoothed aggregation multigrid for Markov chains.* SIAM J. Sci. Comput., 32 (2010), pp. 40-61.

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

[16] P. Dell'Acqua. *Algorithmic variations on the theme of structured matrices, with applications to graphs and imaging.* Ph.D. thesis, Università dell'Insubria (2013).

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

[18] M. Donatelli, M. Semplice, S. Serra-Capizzano. *Analysis of Multigrid preconditioning for implicit PDE solvers for degenerate parabolic equations.* Siam J. Matrix Anal. Appl., 32-4 (2011), pp. 1125–1148.

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

[20] A. Frangioni, C. Gentile. *Prim-based BCT preconditioners for Min-Cost Flow Problems.* Comput. Opt. Appl., 36 (2007), pp. 271–287.

[21] A. Frangioni, C. Gentile. *Experiments with Hybrid Interior Point/Combinatorial Approaches for Network Flow Problems* Optim. Meth. & Softw. 22-4 (2007), pp. 573–585.

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

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

[24] A. Greenbaum. *Analysis of a multigrid method as an iterative technique for solving linear systems.* Siam J. Numerical Anal., 21-3 (1984), pp. 473–485.

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

[26] H.B. Keller. *Numerical Methods for Two-Points Boundary-Value Problems.* Blaisdell, London, 1968.

[27] G. Kirchhoff. *Über die Auflösung der Gleichungen, auf welche man bei der Untersuchung der linearen Verteilung galvanischer Ströme geführt 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.

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

[29] 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.

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

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

[32] Y. Notay. *An aggregation-based algebraic multigrid method.* Electronic Trans. Num. Analysis, 37 (2010), pp. 123–146.

[33] Y. Notay, P. S. Vassilevski. *Recursive Krylov-based multigrid cycles.* Numer. Linear Algebra Appl., 15 (2008), pp. 473–487.

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

[35] 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.

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

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

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

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

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

[41] H.D. Sterck, T.A. Manteuffel, S.F. Mccormick, Q. Nguyen, J. Ruge. *Multilevel Adaptive Aggregation for Markov Chains, with Application to WEB Ranking.* Siam J. Sci. Comput. 30(5), 2235-2262, 2008.

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

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

[44] R.S. Varga. *Matrix Iterative Analysis.* Prentice Hall, Englewood Cliffs, 1962.