On solving symmetric indefinite systems by preconditioned iterative methods

Miroslav Tůma

Institute of Computer Science

Academy of Sciences of the Czech Republic

and Technical University in Liberec

based on joint work with

Michele Benzi

Supported by the project

“Information Society” of the Academy of Sciences of the Czech Republic

under No. 1ET400300415

76th Annual GAMM conference, Luxembourg, March 28 – April 1, 2005

Outline

1. The problem and its motivation
2. Indefinite solvers - an overview
3. Preconditioned iterative methods
4. Matrix preprocessings
5. Numerical experiments
6. Conclusions

1. The problem and its motivation

1. Solving systems with symmetric, generally indefinite matrix

$$Ax = b$$

Arise in many important applications

- Saddle-point problems (CFD, mixed FEM, KKT systems in optimization, optimal control, . . .)
- Helmholtz equation
- “Shift-and-invert”, Jacobi-Davidson algorithms

Efficient solvers and theory exist for some problems (e.g., saddle-point ones. General case seems to be very difficult.

2. Indefinite solvers – an overview

Saddle-point problems: specialized solvers

$$\begin{pmatrix} A & B \\ B^T & -C \end{pmatrix}$$

- Reduction to a definite system
  - ⁎ Schur complement approach
  - ⁎ dual variable (null-space) approach
- Solving original indefinite system
  - ⁎ direct solvers
  - ⁎ preconditioned iterative solvers (block DIAG, block TR, constraint preconditioners, inner block reductions)
- Split and solve approaches (HSS iterations, HSS preconditioners)

2. Indefinite solvers – an overview: II.

Our focus: general indefinite systems

- Sparse direct methods (MA27, MA47, MA57; Duff et al., Pardiso; Schenk and Gärtner)
  - very powerful; have inherent limits of direct methods
- Preconditioned iterative methods
  - SSOR, block SSOR and symmetric ILUT preconditioners (Freund, 1994, 1997)
  - Diagonal pivoting and inverse diagonal pivoting preconditioners; symmetric Krylov methods (Benzi, T., 2002); often useful for weakly indefinite systems
  - Approximate diagonal pivoting decompositions (right-looking, based on linked-lists) for smoothing (Qu, Fish, 2001)
  - Diagonal pivoting preconditioners with diagonal and Bunch-Kaufmann pivoting (left-looking); nonsymmetric Krylov methods (Li, Saad, 2004).
  - Polynomial preconditioners (Saad (1983), Ashby, Manteuffel & Saylor (1989) and Freund (1991).)

3. Preconditioned iterative methods

Our focus, in particular: preconditioned iterative methods

Widely used Krylov methods for symmetric indefinite systems

- MINRES
- SYMMLQ
- simplified QMR

Preconditioning: an example

$$\mathbf{M^{-1}Ax = M^{-1}b} \tag{1}$$

- can be positive definite
- or indefinite
- or even a nonsymmetric solver can be used

Preconditioning: strategies

Positive definite preconditioning

- MINRES and SYMMLQ can be preconditioned, but the transformation is not straightforward, see, e.g., Battermann, 1998.
- Straightforward preconditioning of smoothed CG (equivalent to MINRES)
- Often rather weak

Indefinite preconditioning

- Can be plugged-in in the same way as above; simplified QMR often used as an accelerator
- In practice: Very often useful
- Which of the above algorithms, if any, is theoretically sound? See the talk of Miro Rozložník from Algoritmy 2005.

Preconditioning: theory and practice

- Theoretical insufficiencies reflected in experimental results.
  - The problem is difficult: implementations/algorithms typically very fragile with respect to parameters
  - In many cases, specialized solvers (e.g., reduction based) much better than black-box general indefinite preconditioners
  - Often: Converge fast or never. This effect much more profound than in nonsymmetric or SPD solvers.
- In spite of this: It is worth to develop strategies to precondition symmetric indefinite solvers.

Consider two preconditioner classes

- Incomplete factorizations
- Sparse approximate inverses

Preconditioning: our options

- Right-looking (submatrix) implementation of incomplete decompositions
- Bunch-Parlett-Kaufmann family of pivoting options
  - Bunch-Parlett with various pivotings
  - Bunch-Kaufmann variations
  - Bounded Bunch-Kaufmann (Ashcraft, Grimes, Lewis, 1997)
  - Bunch tridiagonal pivoting (Bunch, 1973; Hagemann, Schenk, 2004), Bunch-Kaufmann pentadiagonal pivoting
  - approximate LDL^T decomposition

Preconditioning: our options (continued)

- Sparse LTL^T decomposition
  - Slow (fill-in in exact case given by $\sum_{i=1}^{n} adj(T[i])$, see Ashcraft, Grimes, Lewis, 1997)
  - if incomplete, large growth in the submatrix
- Saddle-point reconstruction
  - $A = \begin{pmatrix} \hat{A} & \hat{B} \\ \hat{B}^T & -\hat{C} \end{pmatrix}$
  - sometimes useful
  - not much improvement for strongly indefinite problems
- Block diagonal, block symmetric Gauss-Seidel; blocks based on matchings or TPABLO (O'Neil, Szyld,1990); matching-based preprocessings

Bunch-Kaufmann pivoting (Treated in detail in the talk by Olaf Schenk)

Algorithm 1 *Bunch-Kaufmann pivoting strategy for an k-th step of diagonal pivoting decomposition $P^TAP \approx LDL^T$, where D is a block diagonal matrix with blocks of the size 1×1 or 2×2, P is a permutation matrix of the dimension n and L is a block unit lower triangular matrix of the dimension n with blocks conforming to those of D. Parameter α balances 2×2 and 1×1 pivots.*

(1) if $|a_{kk}| \geq \alpha |a_{kl}|$ then use $|a_{kk}|$ as a 1×1 pivot

(2) elseif $|a_{kk} a_{ls}| \geq \alpha a_{kl}^2$ then use $|a_{kk}|$ as a 1×1 pivot

(3) elseif $|a_{ll}| \geq \alpha a_{ls}$ then use $|a_{ll}|$ as a 1×1 pivot

(4) else use the submatrix determined by rows and columns l and k as a 2×2 pivot

Factorized approximate inverse preconditioning (BKSAINV)

Algorithm 2 *Indefinite right-looking BKSAINV algorithm*
Construct the block unit upper triangular matrix $Z \in \mathbb{R}^{n \times n}$ *with* N *column blocks* $Z = [Z_1,\ Z_2, \ldots, Z_N] \in \mathbb{R}^{n \times n}$ *and the block diagonal matrix* $D = diag(D_k)_{k=1,N} \in \mathbb{R}^{n \times n}$ *such that* $Z^T AZ = D$*.*

(1) set $Z = I$*,* $count = 0$
(2) for $k = 1, \ldots, N$
(3) find a pivot block column Z_p *composed from one or two columns which we denote by* z_i *or* z_i, z_{i+1}*, respectively*
(4) if $Z_p \equiv z_i$ *then* $count = count + 1$ *else* $count = count + 2$
(5) $D_p = Z_p^T AZ_p$*, Either* $D_p \in R$ *or* $D_p \in R^{2 \times 2}$*.*
(6) for $j = count + 1, \ldots, n$
(7) $P_{pj} = Z_p^T Az_j$
(8) $z_j = z_j - Z_p D_p^{-1} P_{pj}$
(9) end j
(10) end k

Matrix preorderings: standard nonsymmetric case

- permutation to get a nonzero diagonal –
  a classical technique for nonsymmetric matrices (Duff, 1977)

$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$

Matrix preorderings: standard nonsymmetric case

- permutation to get a nonzero diagonal –
  a classical technique for nonsymmetric matrices (Duff, 1977)

$$\begin{pmatrix} * & & * & & \spadesuit \\ & & * & \spadesuit & \\ * & \spadesuit & & * & \\ & * & \spadesuit & & \\ \spadesuit & & & & \end{pmatrix}$$

4. Matrix preprocessing

Matrix preorderings: standard nonsymmetric case

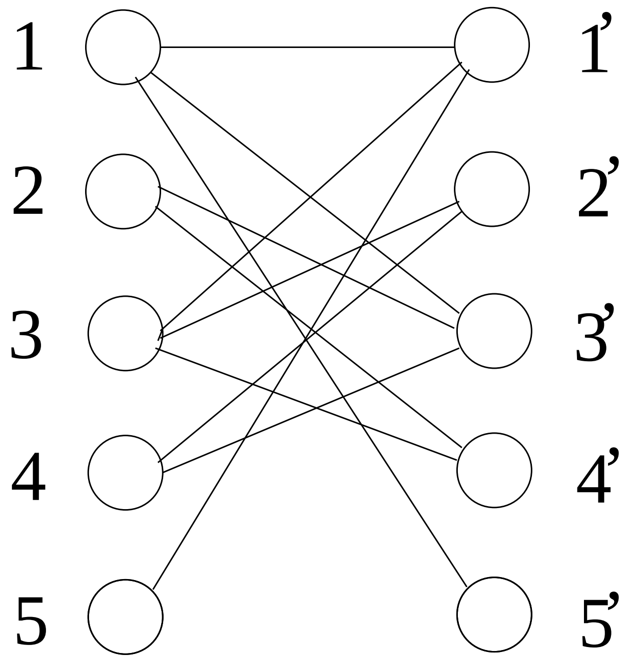
- permutation to get a nonzero diagonal –
  a classical technique for nonsymmetric matrices (Duff, 1977)

$$\begin{pmatrix} * & & * & & \spadesuit \\ & & * & \spadesuit & \\ * & \spadesuit & & * & \\ & * & \spadesuit & & \\ \spadesuit & & & & \end{pmatrix} \qquad \begin{pmatrix} \spadesuit & & & * & * \\ & \spadesuit & & * & \\ & * & \spadesuit & & * \\ & & * & \spadesuit & \\ & & & & \spadesuit \end{pmatrix}$$

Matrix preorderings: standard nonsymmetric case

- permutation to get a nonzero diagonal –
  a classical technique for nonsymmetric matrices (Duff, 1977)

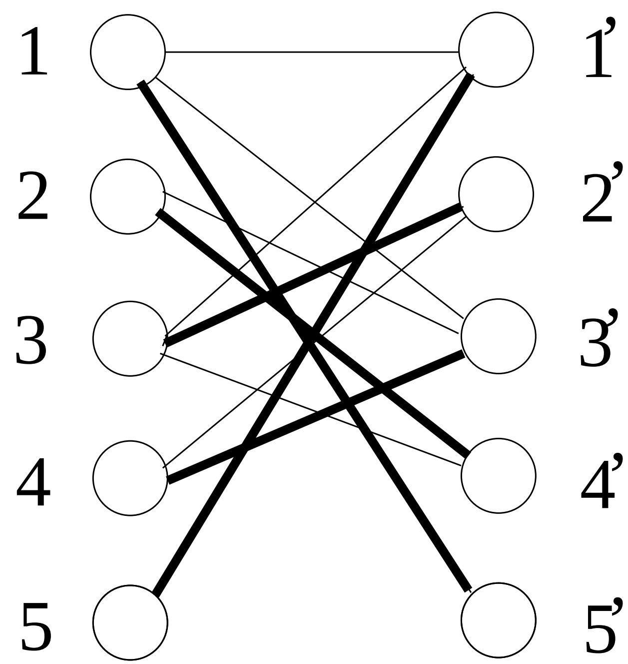
$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$



Matrix preorderings: standard nonsymmetric case

- permutation to get a nonzero diagonal –
  a classical technique for nonsymmetric matrices (Duff, 1977)

$$\begin{pmatrix} * & & * & & \spadesuit \\ & & * & \spadesuit & \\ * & \spadesuit & & * & \\ & * & \spadesuit & & \\ \spadesuit & & & & \end{pmatrix}$$



4. Matrix preprocessing

Matrix preorderings: powerful nonsymmetric case

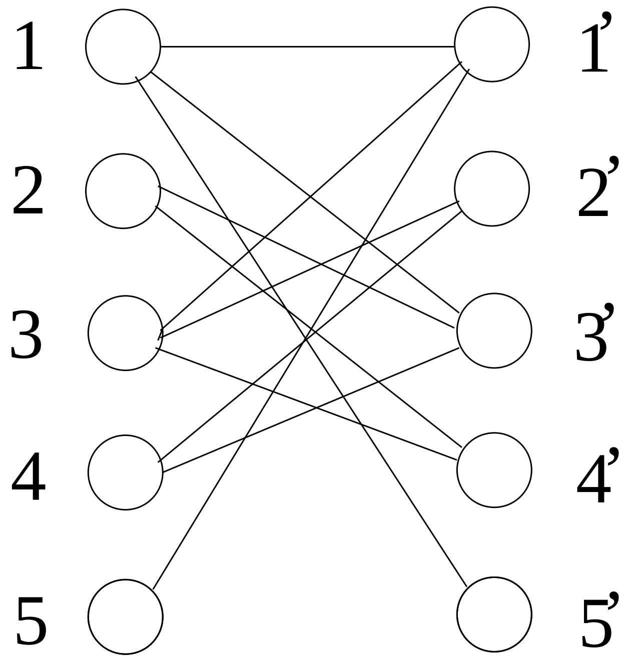
- Using values of matrix entries: strengthening diagonal/block-diagonal dominance
- Useful in both direct solvers and preconditioned iterative solvers
  - e.g. sum/product matching problem – maximize sum/product of modules of transversal entries
  - Olschowka, Neumaier, 1999; Duff, Koster, 1997, 2001; Benzi, Haws, T., 1999.
  - But: permutations are generally nonsymmetric

4. Matrix preprocessing: I.

Matrix preorderings: symmetric case

- start with our example

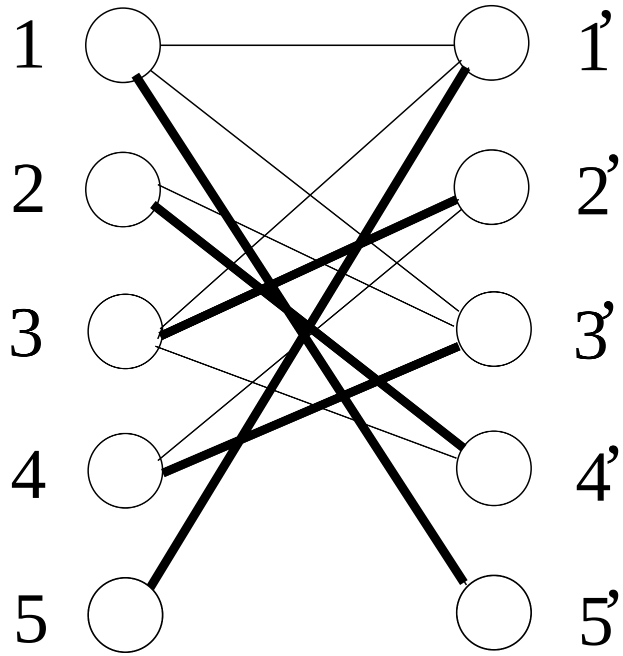
$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$



Matrix preorderings: symmetric case

- start with our example

$$\begin{pmatrix} * & & * & & \spadesuit \\ & & * & \spadesuit & \\ * & \spadesuit & & * & \\ & * & \spadesuit & & \\ \spadesuit & & & & \end{pmatrix}$$

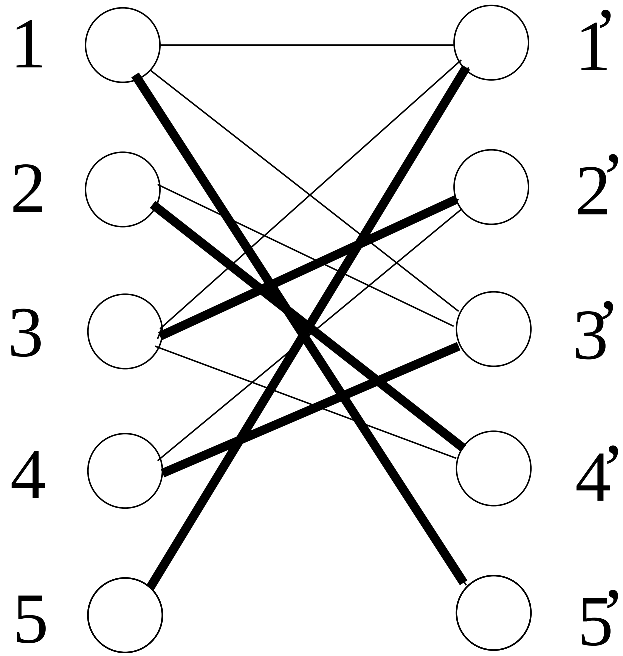


4. Matrix preprocessing: I.

Matrix preorderings: symmetric case

- start with our example

$$\begin{pmatrix} * & & * & & \spadesuit \\ & & * & \spadesuit & \\ * & \spadesuit & & * & \\ & * & \spadesuit & & \\ \spadesuit & & & & \end{pmatrix}$$

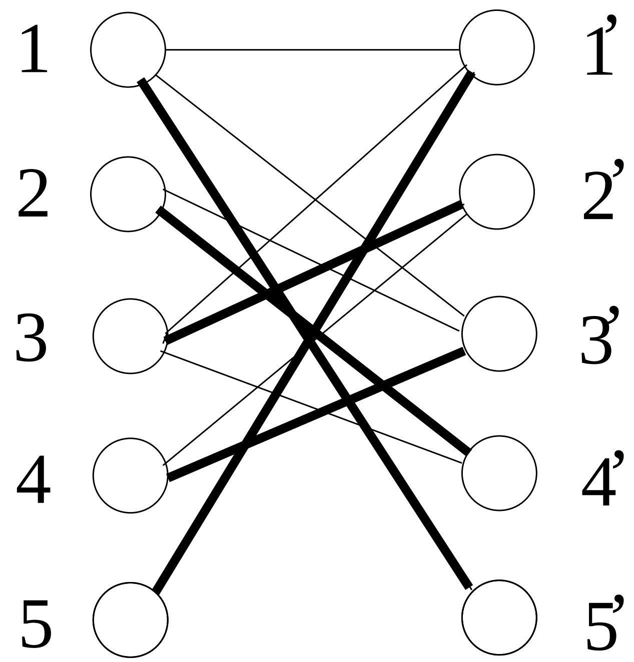


But the permutation is nonsymmetric

Matrix preorderings: symmetric case

- start with our example

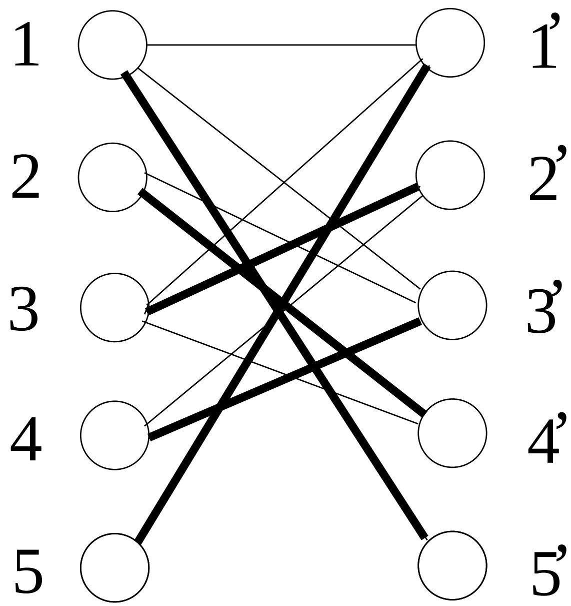
$$\begin{pmatrix} * & & * & & \spadesuit \\ & & * & \spadesuit & \\ * & \spadesuit & & * & \\ & * & \spadesuit & & \\ \spadesuit & & & & \end{pmatrix}$$

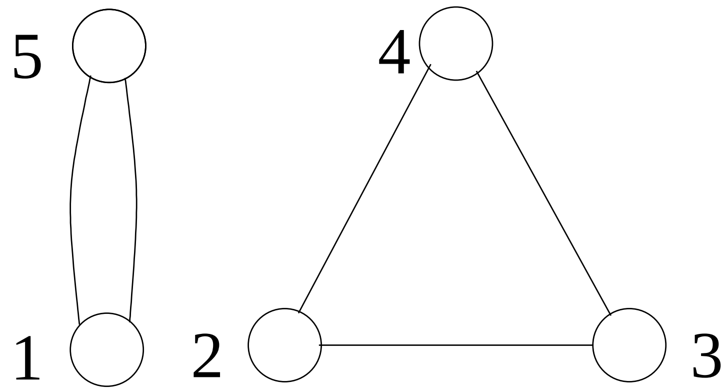


But the permutation is nonsymmetric

Idea of Iain Duff and John Gilbert (2002) – split the loops of a nonsymmetric permutation

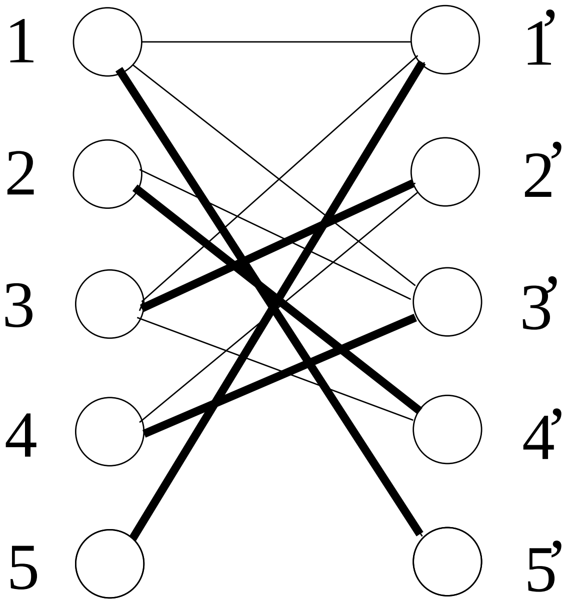
Matrix preorderings: symmetric case: symmetrization based on loops

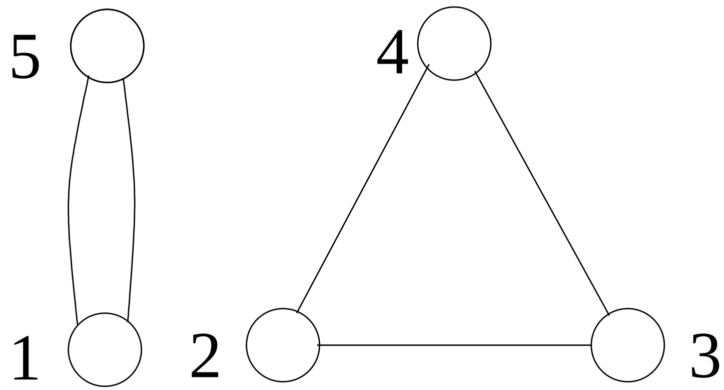




4. Matrix preprocessing: II.

Matrix preorderings: symmetric case: symmetrization based on loops

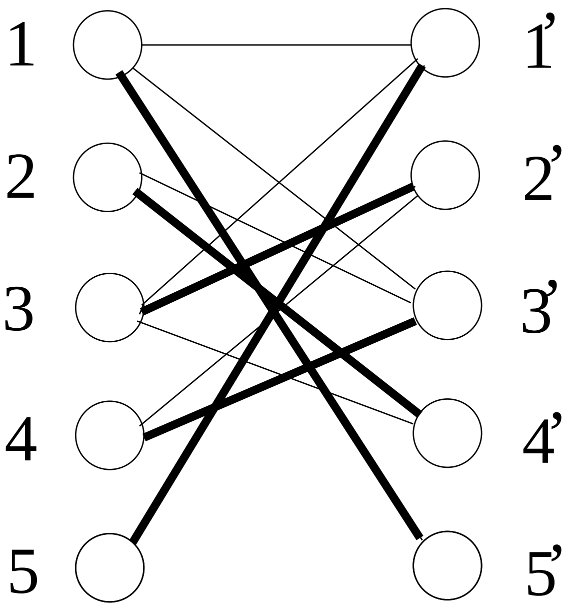


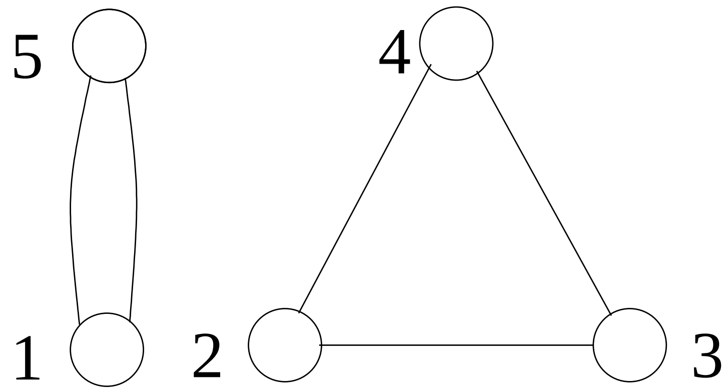


$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix} \longrightarrow \begin{pmatrix} * & * & & * & \\ * & & & & \\ & & & * & * \\ * & & * & & * \\ & & * & * & \end{pmatrix}$$

4. Matrix preprocessing: II.

Matrix preorderings: symmetric case: symmetrization based on loops





$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix} \longrightarrow \begin{pmatrix} \boxed{0} & \boxed{*} & & * & \\ \boxed{*} & \boxed{0} & & & \\ & & \boxed{0} & \boxed{*} & * \\ * & & \boxed{*} & \boxed{0} & * \\ & & * & * & \boxed{0} \end{pmatrix}$$

4. Matrix preprocessing: III.

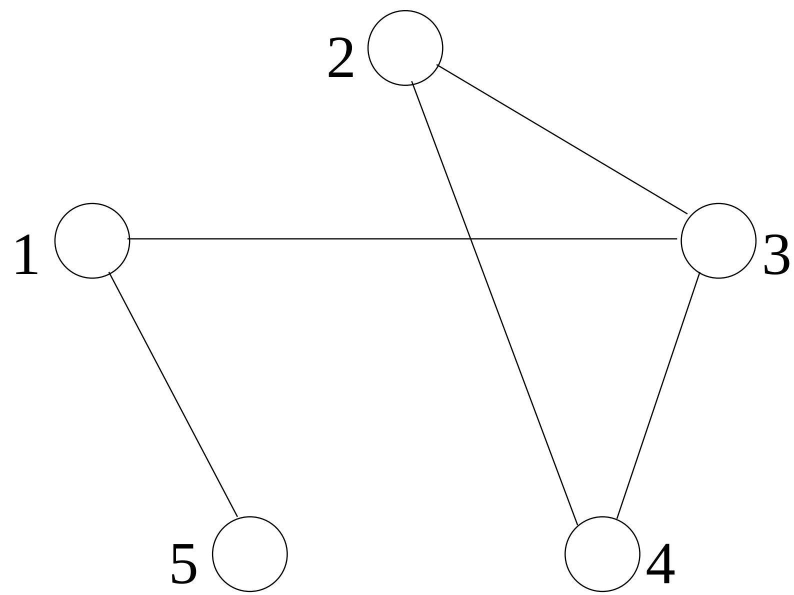
Matrix preorderings: symmetric case: symmetrization based on loops

- Summarized idea:
  - ∗ bipartite matching → nonsymmetric permutation
  - ∗ loops → general matching
- Previous work based on this strategy:
  - ∗ static preordering for direct methods: Duff, Pralet, 2004; additional criterion: based on sparsity of rows/columns
  - ∗ preordering for approximate decompositions for preconditioning : Hagemann, Schenk, 2004

4. Matrix preprocessing: IV.

Matrix preorderings in symmetric case: new idea: general graph matching

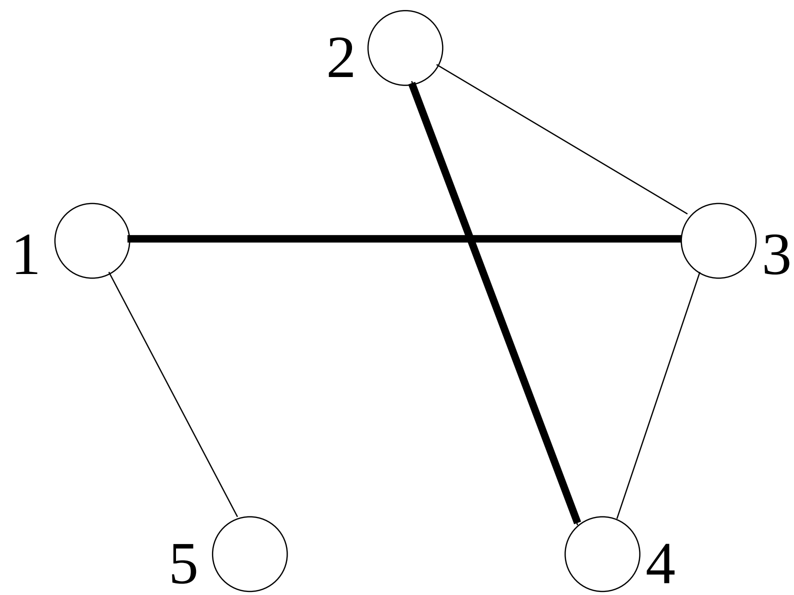
$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$



4. Matrix preprocessing: IV.

Matrix preorderings in symmetric case: new idea: general graph matching

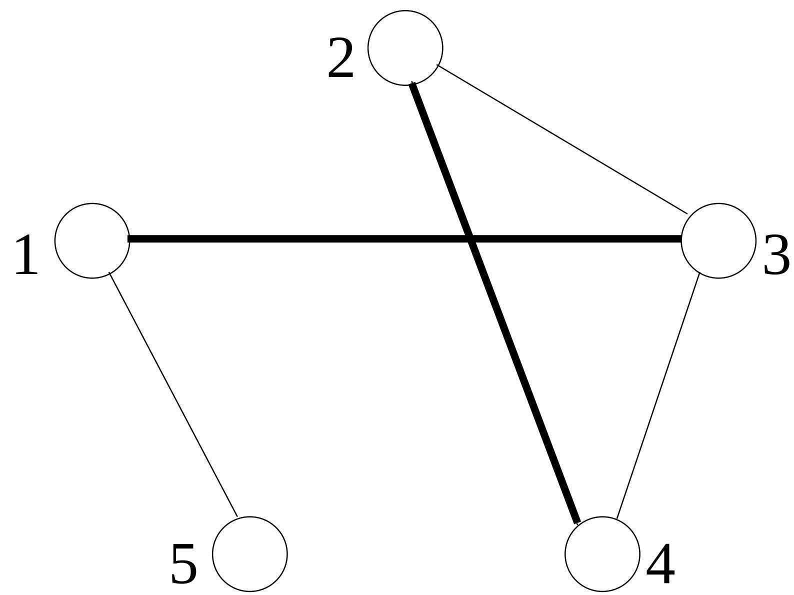
$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$



4. Matrix preprocessing: IV.

Matrix preorderings in symmetric case: new idea: general graph matching

$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$

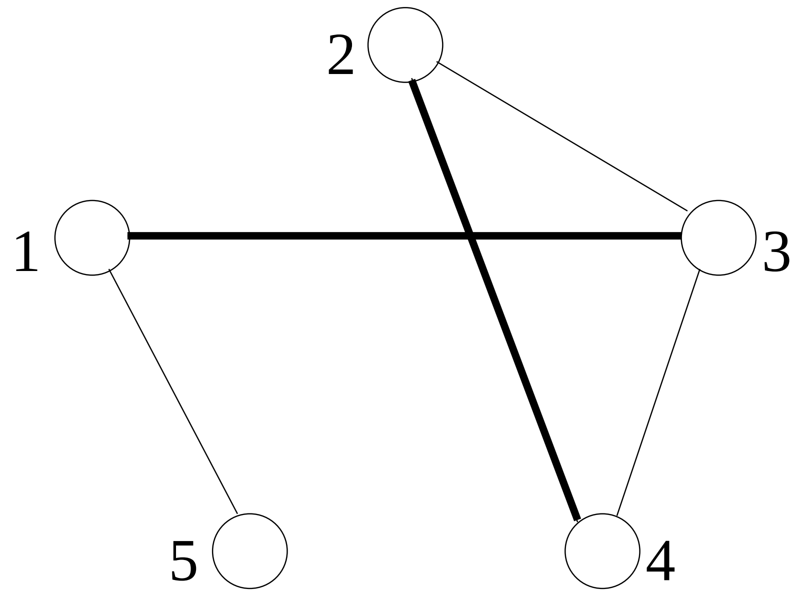


- avoids problems with splitting odd loops

4. Matrix preprocessing: IV.

Matrix preorderings in symmetric case: new idea: general graph matching

$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$



- avoids problems with splitting odd loops
- how to define graph edge weights?

4. Matrix preprocessing: V.

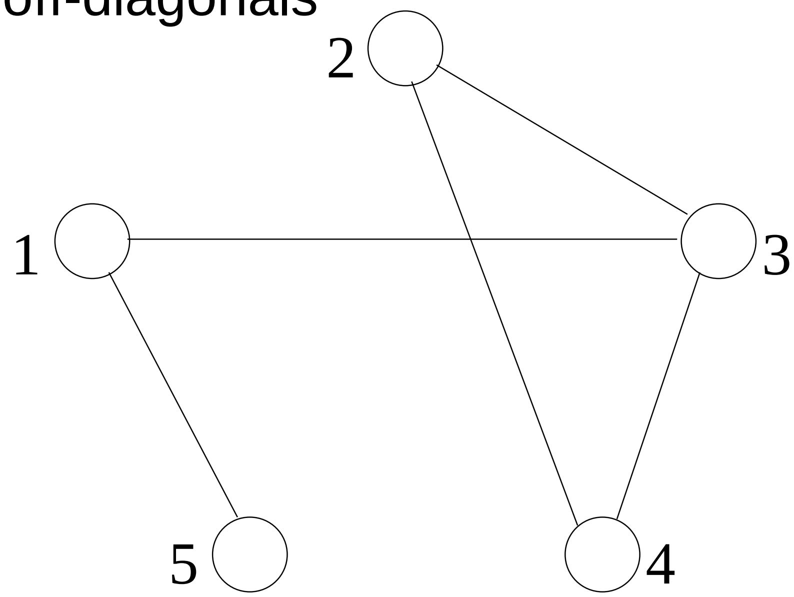
How to define graph weights

- first possibility:

$$weight_{ij} = |a_{ij}| + \alpha(|a_{ii}| + |a_{jj}|)$$

- α balances influence of diagonals and off-diagonals

$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$



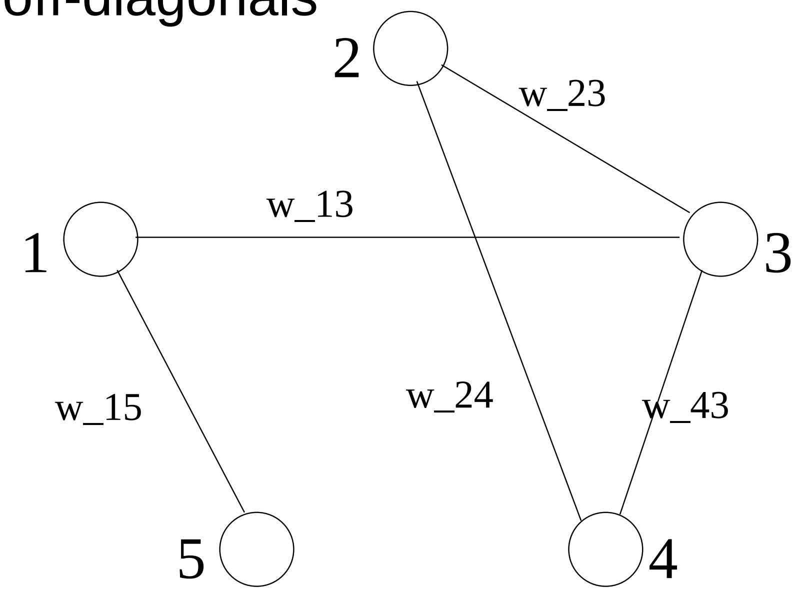
How to define graph weights

- first possibility:

$$weight_{ij} = |a_{ij}| + \alpha(|a_{ii}| + |a_{jj}|)$$

- α balances influence of diagonals and off-diagonals

$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$



4. Matrix preprocessing: V.

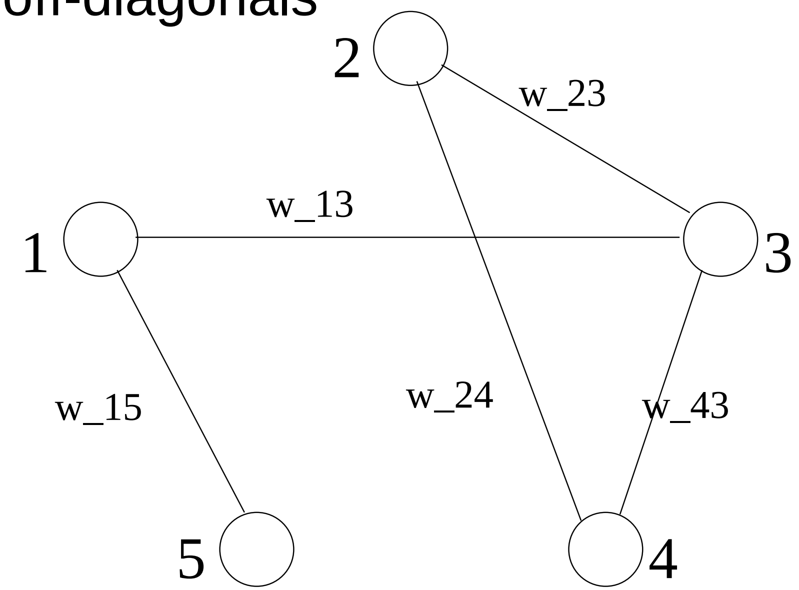
How to define graph weights

- first possibility:

$$weight_{ij} = |a_{ij}| + \alpha(|a_{ii}| + |a_{jj}|)$$

- α balances influence of diagonals and off-diagonals

$$\begin{pmatrix} * & & * & & * \\ & & * & * & \\ * & * & & * & \\ & * & * & & \\ * & & & & \end{pmatrix}$$

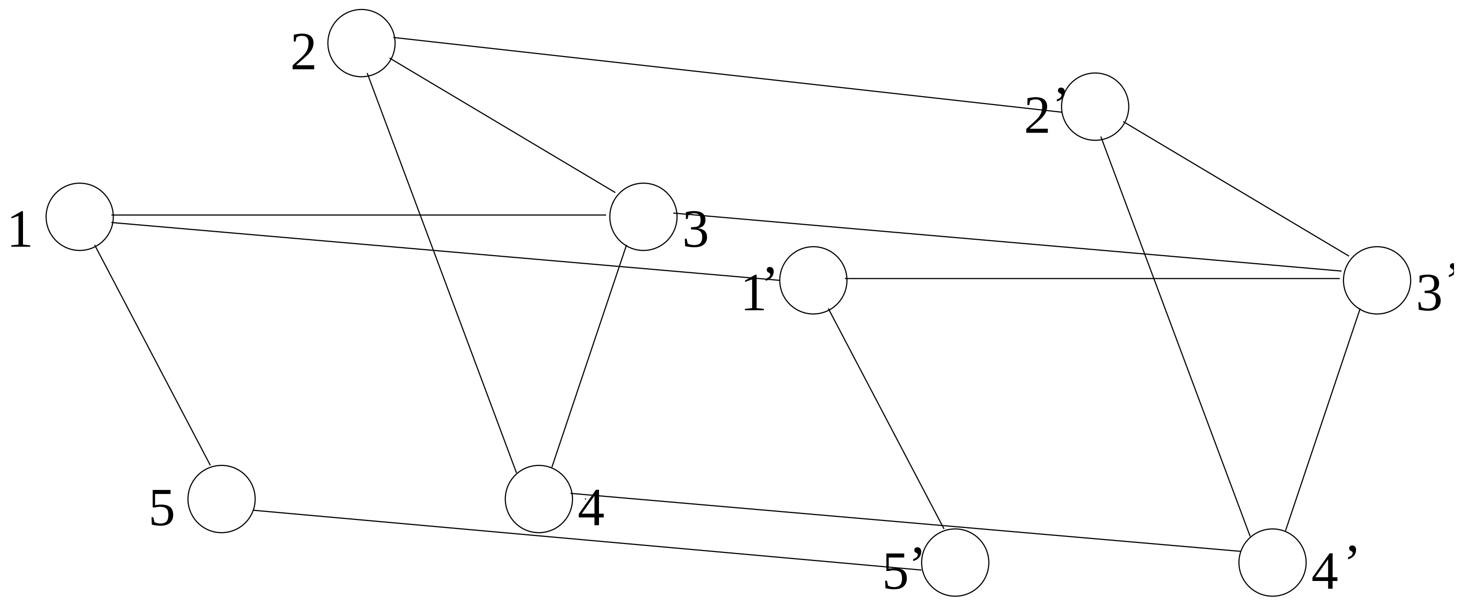


general weighted matching $+$ block construction

4. Matrix preprocessing: V.

Another way to define graph weights

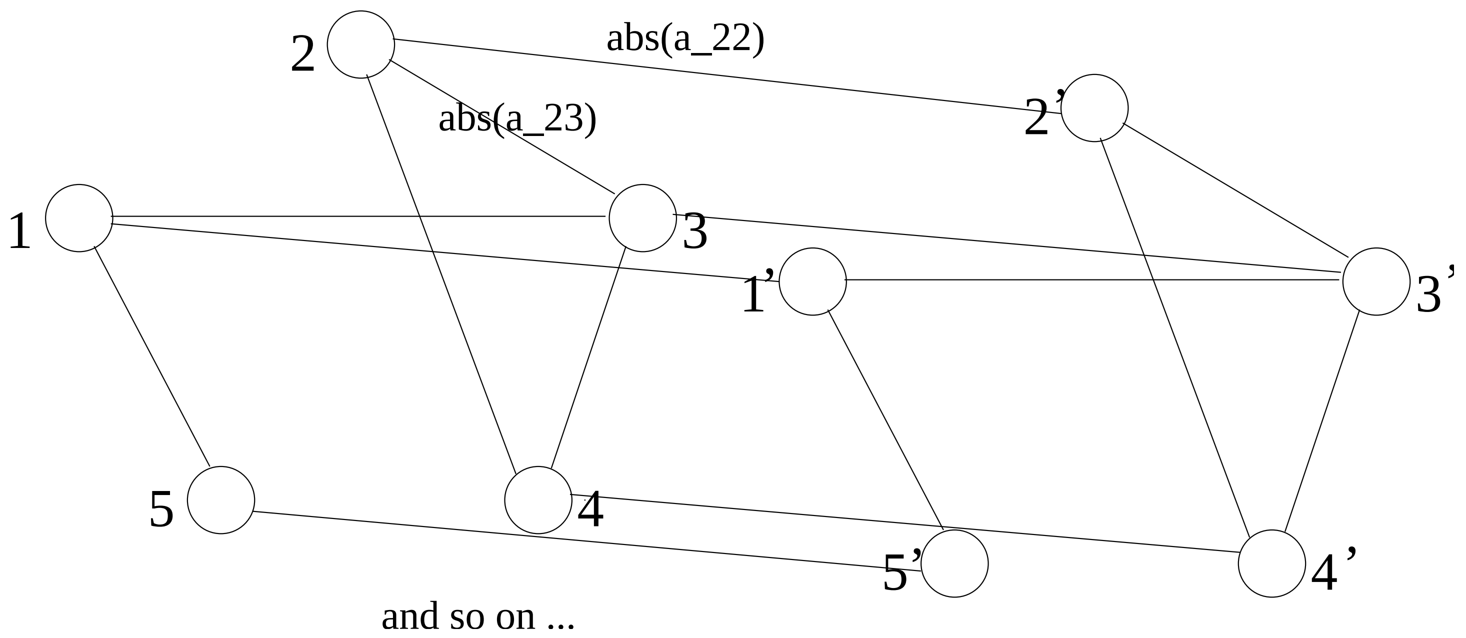
- derived (doubled) graph



4. Matrix preprocessing: V.

Another way to define graph weights

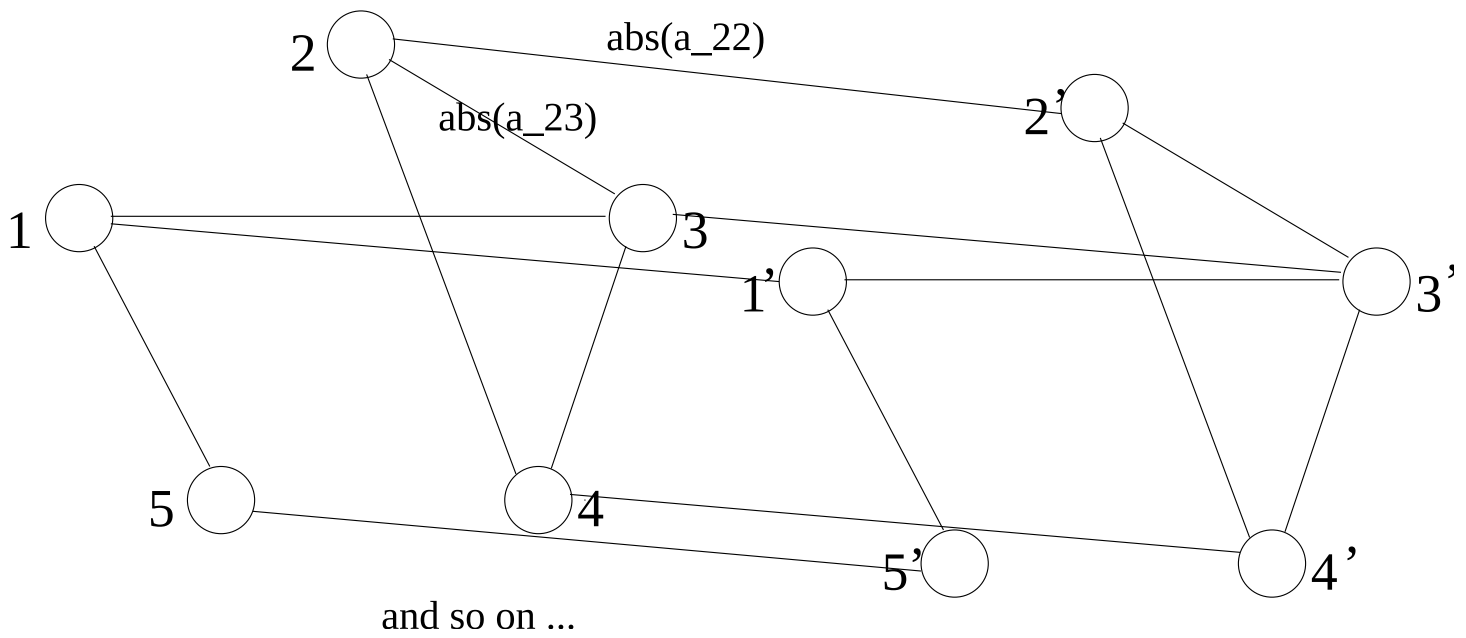
- derived (doubled) graph



4. Matrix preprocessing: V.

Another way to define graph weights

- derived (doubled) graph

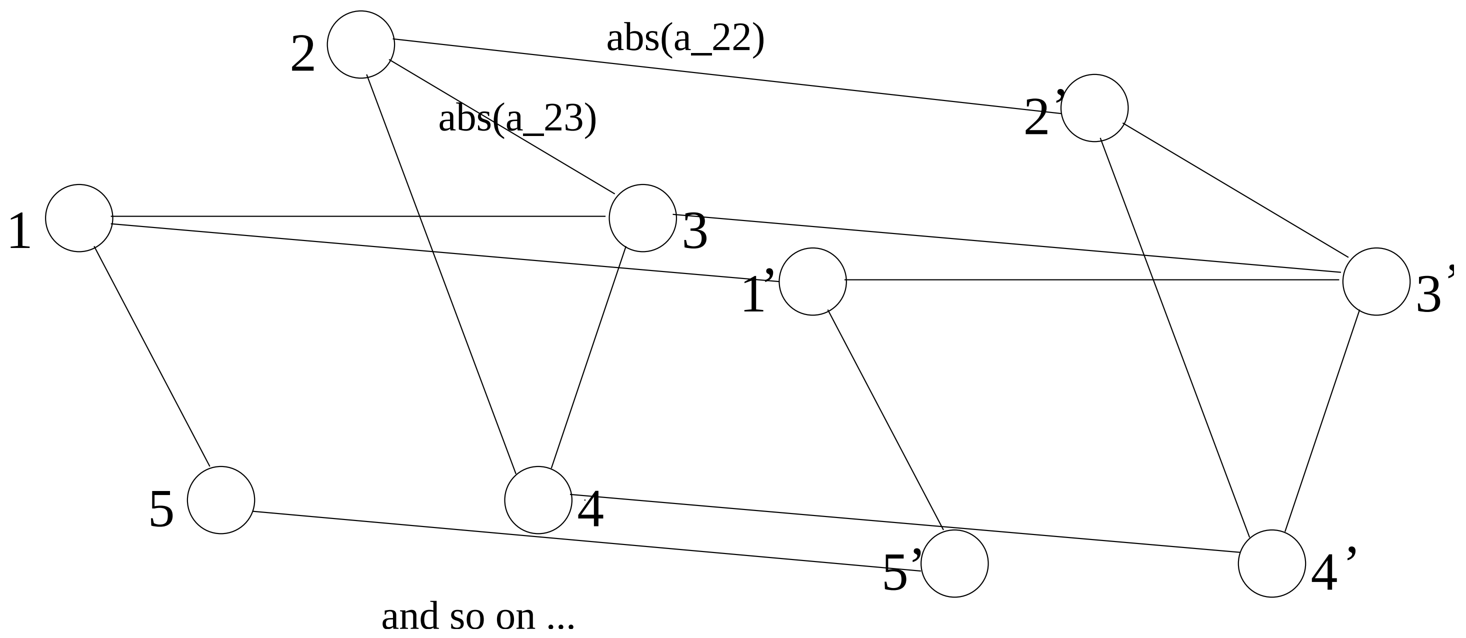


- enables better separation of 1×1 and 2×2 blocks

4. Matrix preprocessing: V.

Another way to define graph weights

- derived (doubled) graph



- enables better separation of 1×1 and 2×2 blocks
- more time-consuming

5. Experimental results

Some parameters of experiments

- preconditioned minimum residual method (implemented as smoothed CG;
  similar results for preconditioned simplified QMR)
- MMD on blocks/vertices
- Results with incomplete decompositions; tridiagonal pivoting in most cases; whenever this fails, replaced by Bunch-Kaufmann pivoting
- bipartite matching by MC64 (Duff, Koster, 2001; HSL)
- general matching by a greedy heuristic (tested also Blossom 3 (Cook, Rohe, 1998); SMP (Burkard, Derigs, 1980); WMATCH (Rothberg, 1973)).
- in some cases: other preprocessings are better, e.g., TPABLO (O'Neil, Szyld, 1990)
- stopping criterion: relative residual norm reduction by 10^{-8}
- a subset of matrices from Li, Saad (2004); Hagemann, Schenk (2004)

5. Experimental results

Tested matrices

$Matrix$	n	nz
C-41	9769	55757
C-19	2327	12072
C-64	51035	384438
C-70	68924	363955
C-71	76638	468096
traj33	20006	504090
traj27	17148	242286
stiff5	33410	177384
mass05	33410	241140
heat02	10295	90129

5. Experimental results

Some results of comparison of preprocessings

	symm matching		general matching		no matching	
$Matrix$	$Size\_p$	its	$Size\_p$	its	$Size\_p$	its
C-41	118323	315	117168	129	100648	23
C-19	26411	7	26805	8	27833	5
C-64	604274	63	615061	55	493287	184
C-70	1186138	12	1162256	9	865192	8
C-71	1421761	15	1421994	45	1388772	117
traj33	221223	464	102629	186	102024	146
traj27	106633	471	105819	140	104679	153
stiff5	202983	80	217119	72	287761	166
mass05	41817	24	41216	52	56865	$\dagger$
heat02	247912	34	410773	45	631236	53

6. Conclusions

Conclusions and future work

- Incomplete diagonal pivoting preconditioning can help in many cases
  - remind that it is very fragile
  - some problems (shifted Laplacians) still a challenge
- Block preprocessing techniques can improve the behavior of solvers; the techniques should be developed further
- All preprocessings: still not mature
- Solving general indefinite systems is very difficult: in many cases only one of more specialized techniques (Schur complement approach, dual variable approach) works