

Maximum-weighted matching strategies and the application to symmetric indefinite systems

Stefan Röllin

Integrated Systems Laboratory, Swiss Federal Institute of Technology, ETH Zurich, Switzerland

roellin@iis.ee.ethz.ch

Olaf Schenk

Department of Computer Science Department, University Basel, Switzerland

olaf.schenk@unibas.ch



Motivation

References to investigations on [unsymmetric matrix permutations](#):

- Olschowka/Neumaier [’96_{LAA}] **A New Pivoting Strategy for Gaussian Elimination**
- Duff/Koster [’99_{SIMAX}] **The design and use of algorithms for permuting large entries to the diagonal of sparse matrices**
- Benzi/Haws/Tuma [’00_{SISC}] **Preconditioning highly indefinite and nonsymmetric matrices**
- Duff/Koster [’01_{SIMAX}] **On Algorithms For Permuting Large Entries to the Diagonal of a Sparse Matrix**
- Schenk/Röllin/Gupta [’04_{IEEE TCAD}] **The Effects of Unsymmetric Matrix Permutations and Scalings in Semiconductor Device and Circuit Simulation**

What about symmetric indefinite systems?

Applications: constrained optimization, saddle-point, eigenvalue or Stokes problems, ...

Maximum-weighted bipartite matchings

Goal: find permutation of rows, such that **product of diagonal elements is maximal**

⇒ bipartite matching

$$\tilde{A} = P_r D_r A D_c$$

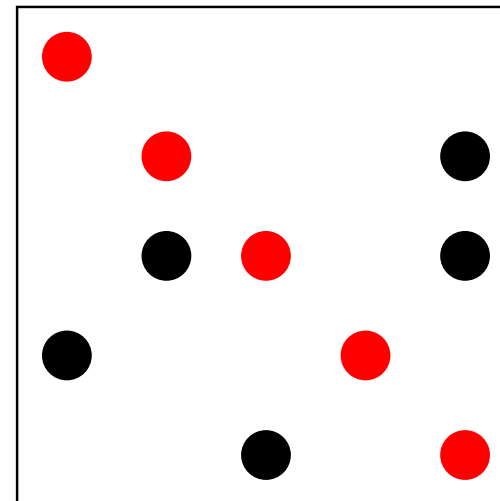
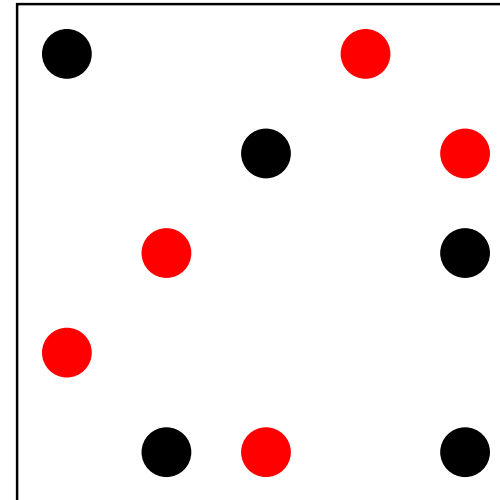
(I-Matrix)

$$|\tilde{a}_{ii}| = 1$$

$$|\tilde{a}_{ij}| \leq 1$$

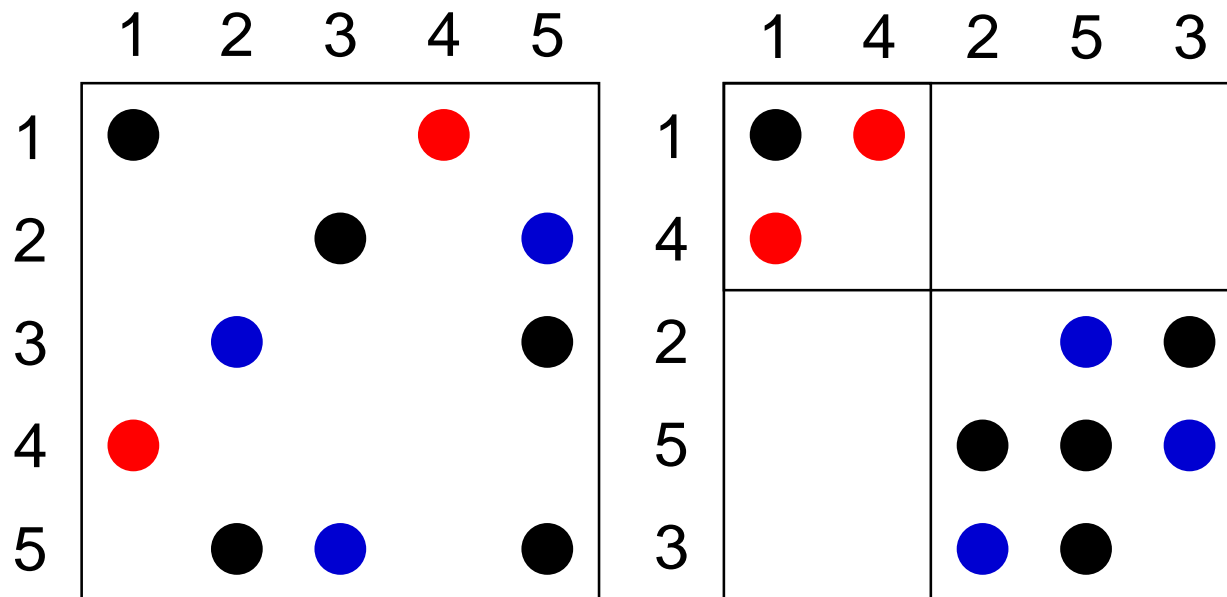
- less pivoting required
- diagonal pivoting **more stable**
- iterative methods **more robust**

Algorithm: MPS (similar to MC64)

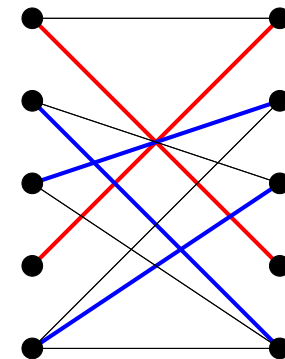


Extension to symmetric matrices

Goal: Find symmetric permutation, such that **large elements** lie in **diagonal blocks** instead of diagonal



Diagonal blocks correspond to cycles of permutation P_r



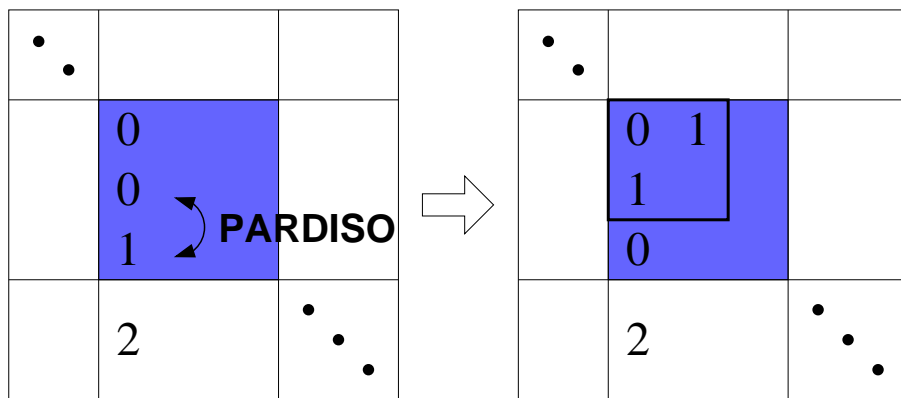
Why do these diagonal blocks help?

Duff/Gilbert [’02_{Householder Symposium}] **Symmetric Weighted Matching and Block Pivoting for Symmetric Indefinite Systems**

Block Bunch-Kaufman pivoting in PARDISO

Sparse Gaussian Elimination with Bunch/Kaufmann pivoting

- PARDISO Symmetric indefinite direct solver (O. Schenk, 2003)
 - ability of doing numerical symmetric pivots using 2×2 blocks in **supernodes**
 - Level-3 BLAS factorization $A = QLDL^T Q^T$ with Bunch/Kaufmann supernode pivoting



The growth of the diagonal elements is controlled with:

```
if ( $|d_{ii}| < \epsilon$ ) then  
    set  $d_{ii} = \text{sign}(d_{ii}) \cdot \epsilon$   
endif
```

⇒ Force diagonal blocks to be in supernodes

Long cycles

Problem: rows/columns in **diagonal blocks** possibly have **different structures**
⇒ **high fill-in** during factorization

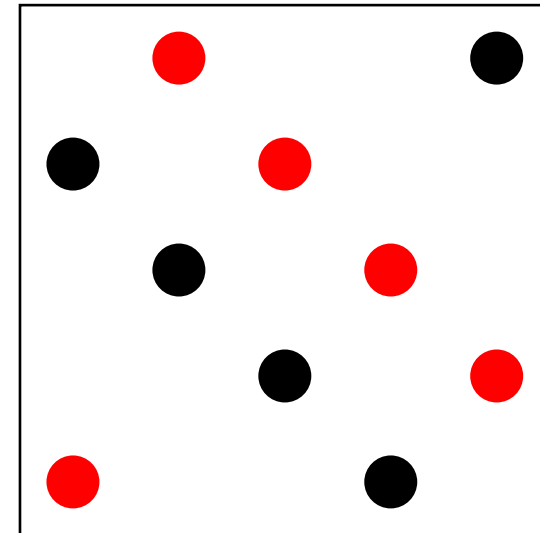
Solution: Split large diagonal blocks into smaller blocks

Split **blocks of even size**:

- into blocks of size **two**

Split **blocks of odd size**:

- into blocks of size **two** and **three**
- into blocks of size **two** and **one**



Duff/Pralet [’04_{SIAM Workshop}] **Symmetric Weighted Matching and Application to Indefinite Multifrontal Solvers**

Long cycles

Problem: rows/columns in **diagonal blocks** possibly have **different structures**
⇒ **high fill-in** during factorization

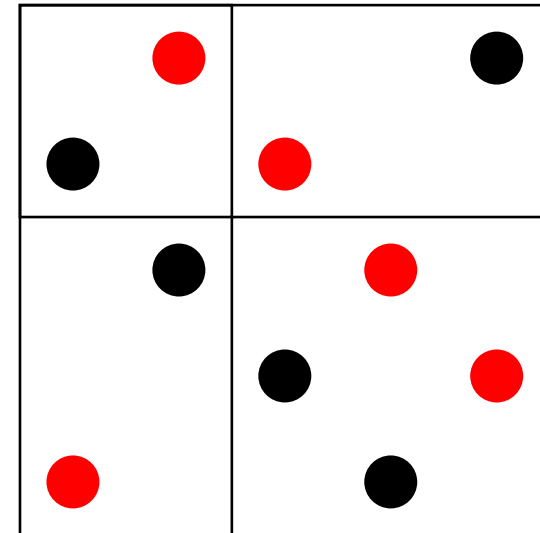
Solution: Split large diagonal blocks into smaller blocks

Split **blocks of even size**:

- into blocks of size **two**

Split **blocks of odd size**:

- into blocks of size **two** and **three**
- into blocks of size **two** and **one**



Duff/Pralet [’04_{SIAM Workshop}] **Symmetric Weighted Matching and Application to Indefinite Multifrontal Solvers**

Long cycles

Problem: rows/columns in **diagonal blocks** possibly have **different structures**
⇒ **high fill-in** during factorization

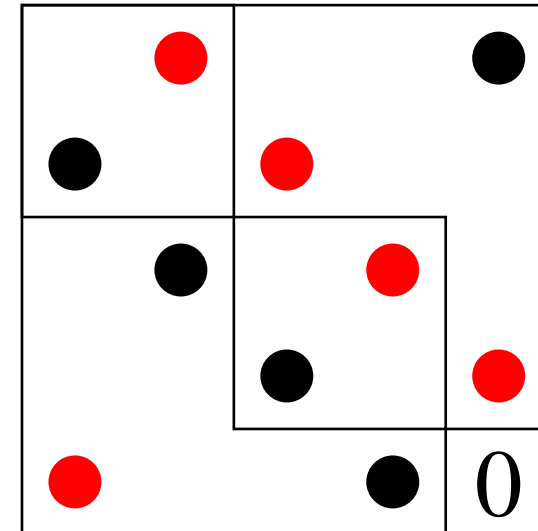
Solution: Split large diagonal blocks into smaller blocks

Split **blocks of even size**:

- into blocks of size **two**

Split **blocks of odd size**:

- into blocks of size **two** and **three**
- into blocks of size **two** and **one**



Duff/Pralet [’04_{SIAM Workshop}] **Symmetric Weighted Matching and Application to Indefinite Multifrontal Solvers**

Preprocessing strategies

Four strategies for diagonal blocks ([without scaling](#)):

default	no preprocessing
full blocks	keep large diagonal blocks
2x2/1x1	divide blocks larger than two in blocks of size one and two
2x2/3x3	break into blocks of size two and three

For unsymmetric matrices: scaling to I-Matrix possible

Symmetric case: $\tilde{A} = PDADP^T$, where $D = \sqrt{D_r D_c}$

- $|\tilde{a}_{ij}| \leq 1$
- matched elements in diagonal blocks ($|\cdot| = 1$)

⇒ four additional strategies [with scaling](#)

Overall strategy, test environment

1. do **preprocessing**
 - find unsymmetric permutation
 - extract cycles (diagonal blocks)
 - split them up
(depending on preprocessing)
2. compute **symmetric** (block) **permutation** to reduce fill-in
3. compute symbolic and numerical **factorization**
4. do **solve**
(iterative refinement if necessary)

60 symmetric indefinite matrices

Gould/Scott [’03_{TR-2003-19}] **A numerical evaluation of HSL packages for the direct solution of large sparse, symmetric linear systems of equations**

Two scenarios:

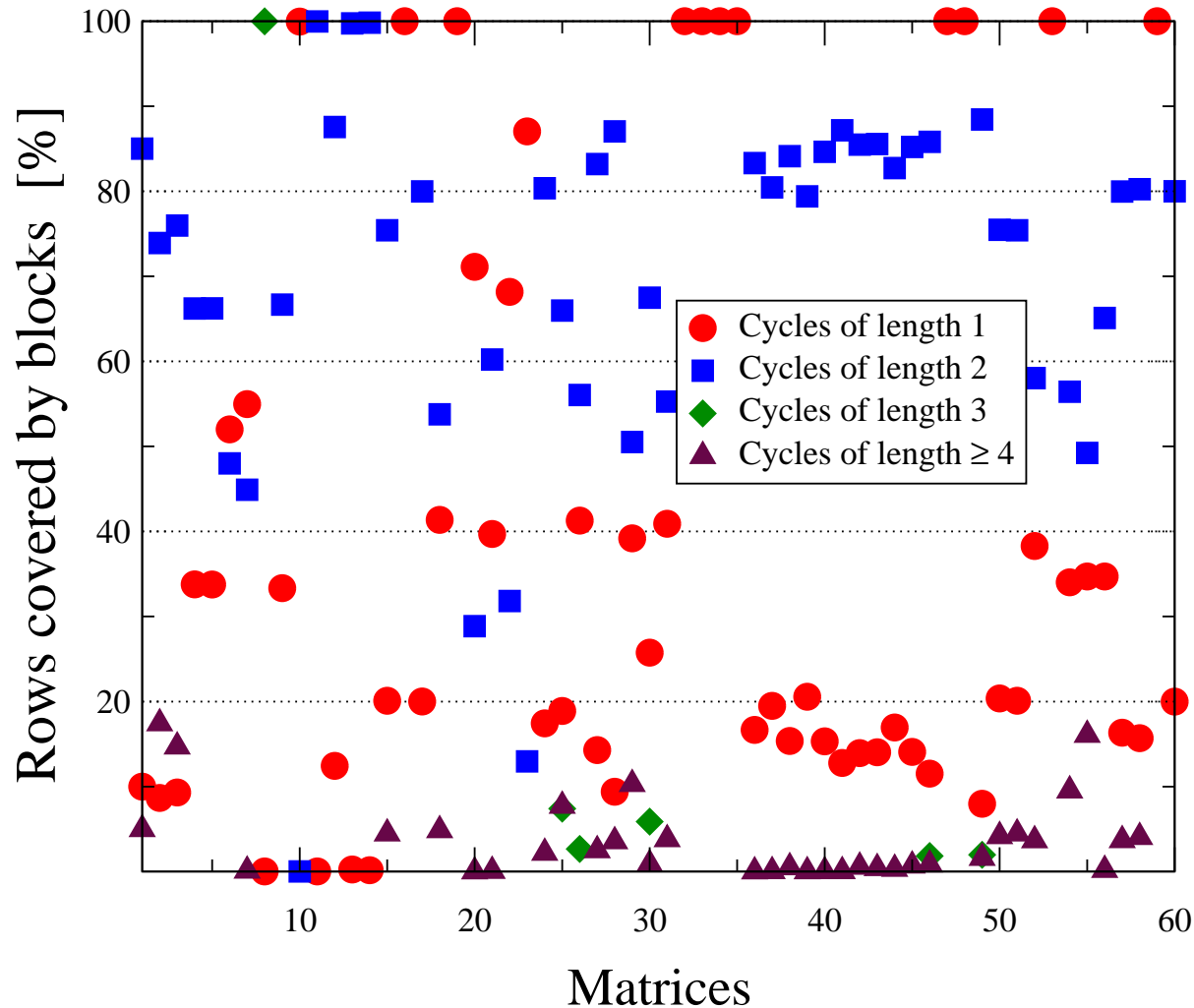
- **Standard accuracy**

with one step of iterative refinement
 $\|b - Ax\| / (\|A\|\|x\| + \|b\|) < 10^{-4}$
and iterative refinement converges

- **Extended accuracy**

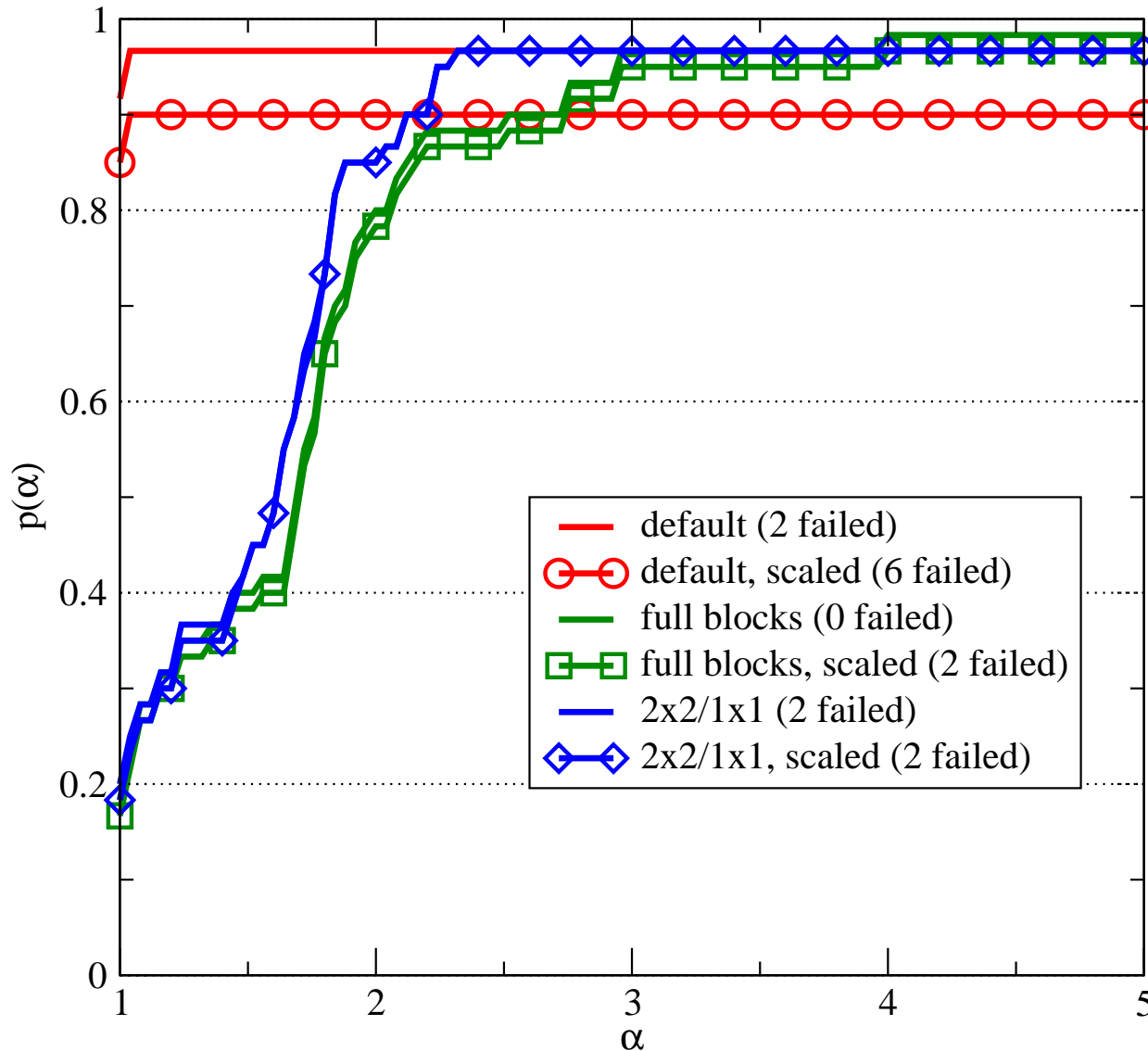
without iterative refinement
 $\|b - Ax\| / (\|A\|\|x\| + \|b\|) < 10^{-11}$
and iterative refinement converges

Length of Cycles



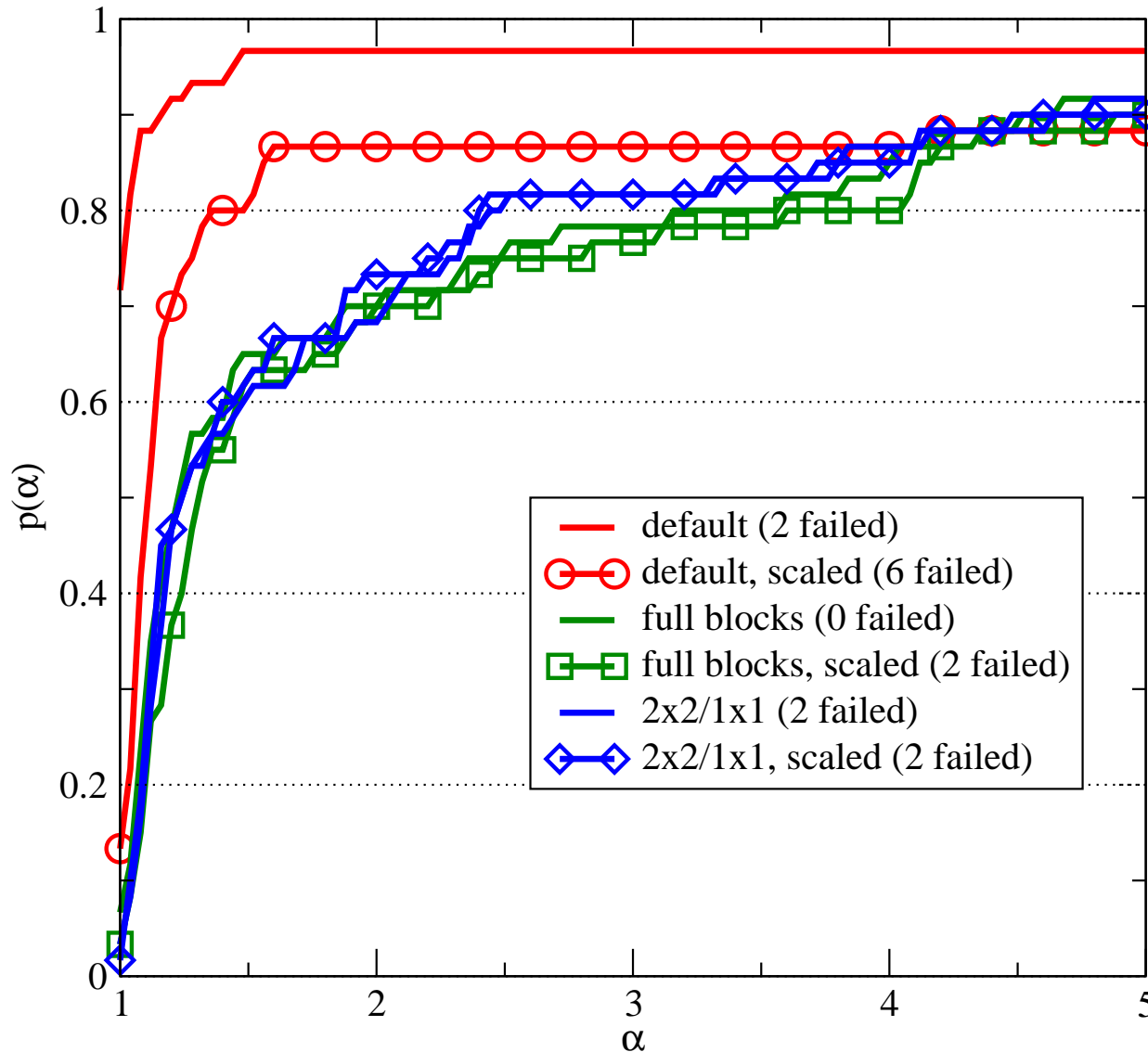
- most blocks of size 1 and 2
24 matrices blocks ≤ 2
- only 7 with odd blocks
- 38 matrices have
less than 1% blocks > 2

Standard accuracy - memory for factorization



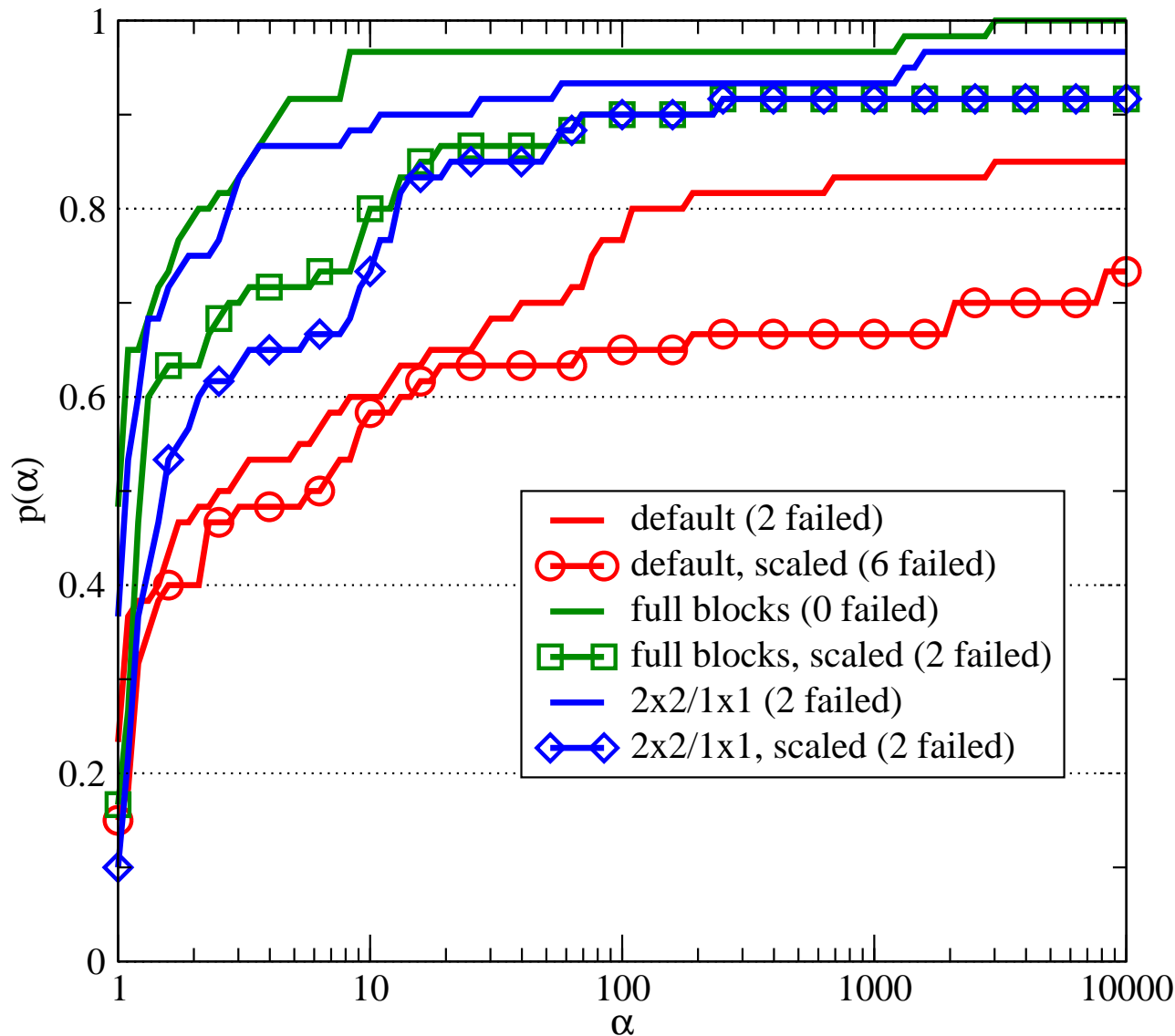
- **performance profile:** shows fraction $p(\alpha)$ of examples, which can be solved within α times best strategy
- **number of nonzeros** required for factorization
- **default** requires fewest memory
- **2x2/1x1** needs less memory than **full blocks**

Standard accuracy - total time



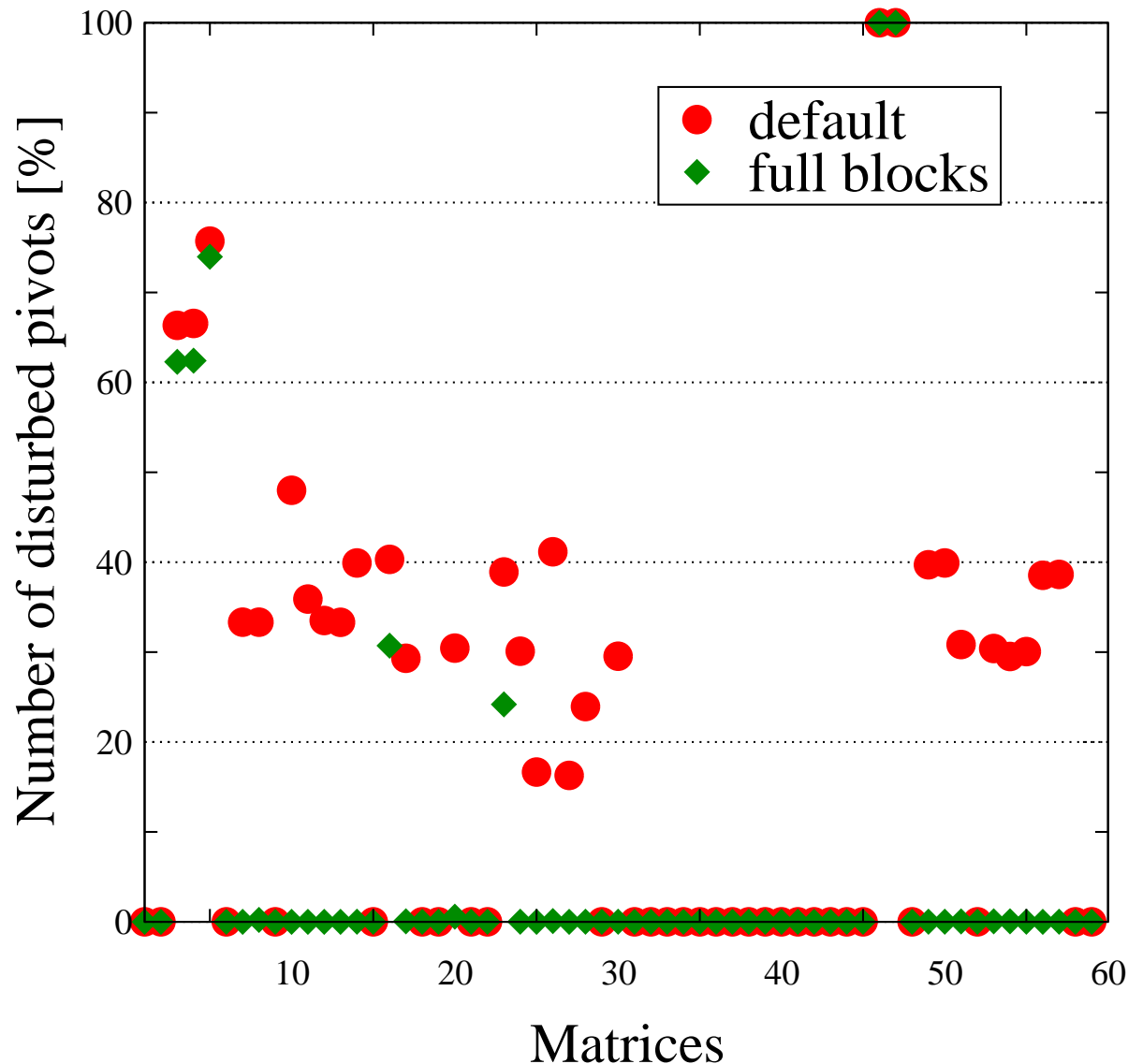
- **Time** for matching, analyze, factorization and solve
- Similar to memory requirements
- Computation of preprocessing cannot be neglected (for 80% within 1.5 of default)

Standard accuracy - first scaled residual



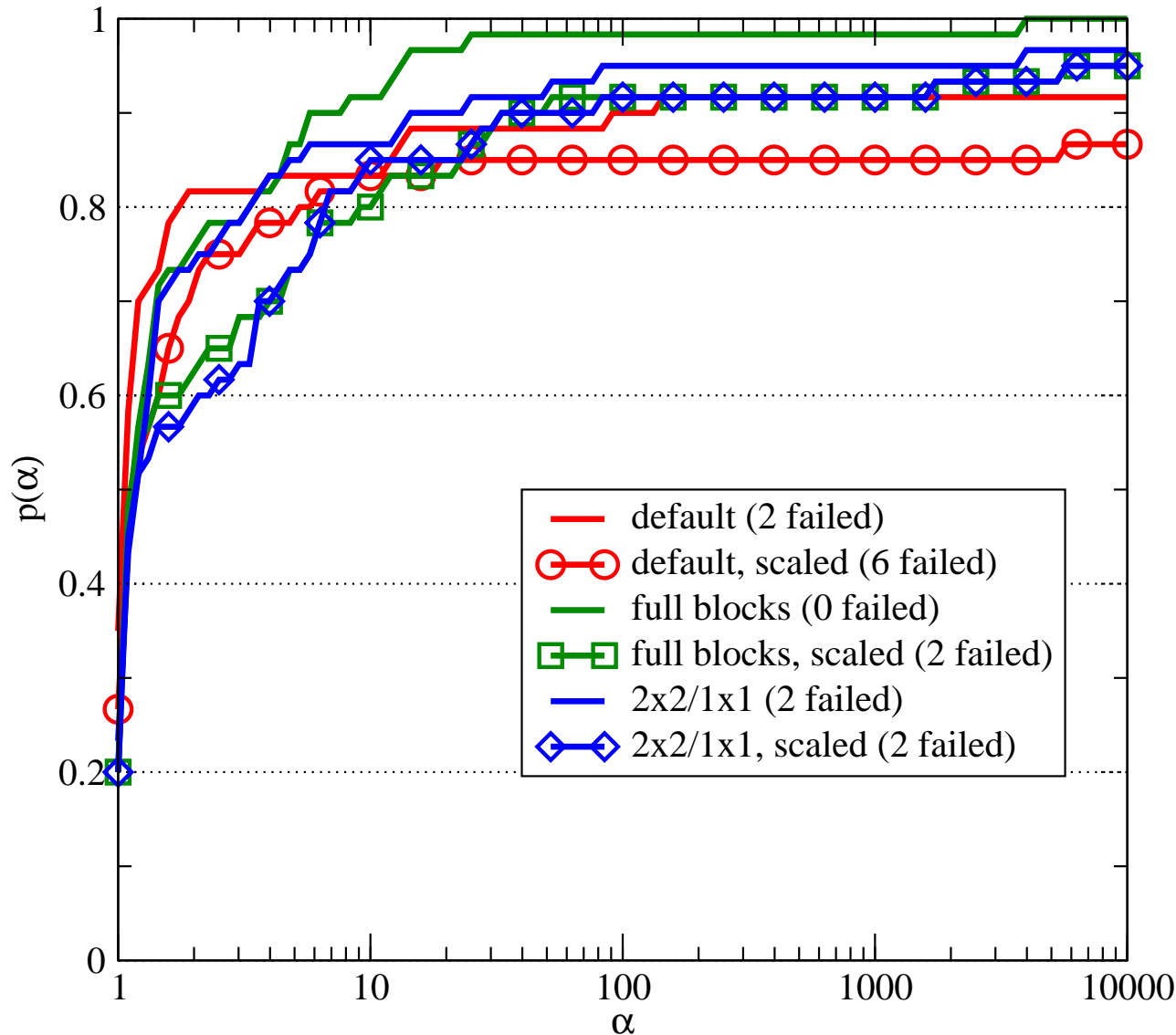
- Depicted is **accuracy of scaled residual:**
 $\|b - Ax\| / (\|A\|\|x\| + \|b\|)$
without iterative refinement
- Using a preprocessing strategy improves accuracy compared to no preprocessing
- **Full blocks** gives best results followed by **2x2/1x1**
- Scaling the matrices deteriorates accuracy

Perturbed pivots



- Number of **perturbed pivots** during factorization
- For 30% of matrices no perturbed pivots required
- Significant reduction for **full blocks** (and other strategies): only 7 matrices more than 1% perturbed pivots
- Scaling the matrices gives similar results

Standard accuracy - second scaled residual

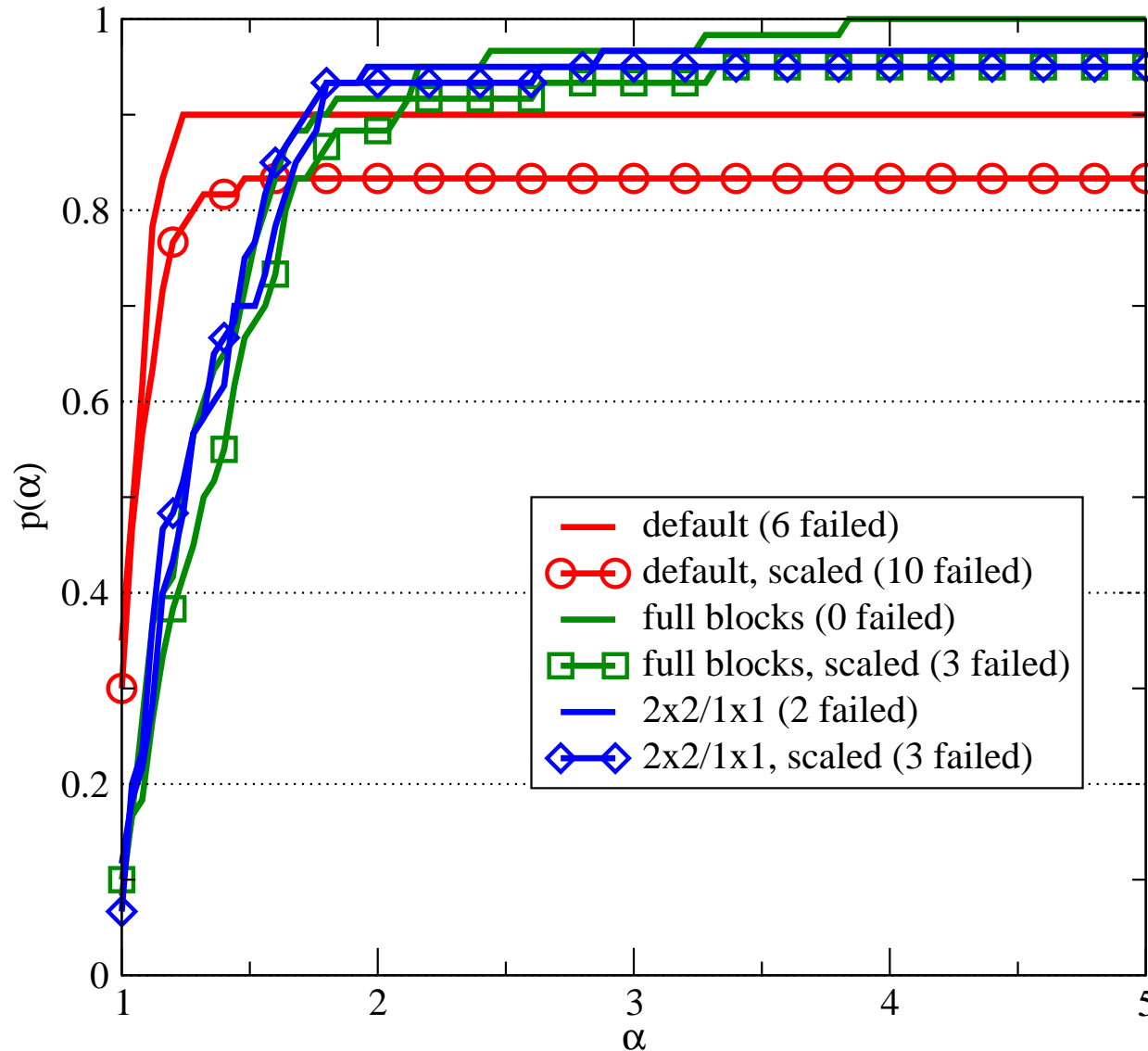


- **Scaled residuals** after one step iterative refinement
- Fewer differences between strategies
- **Full blocks** still slightly better than other strategies
- Strategies with scalings still worse than without scalings

Extended accuracy

Strategy	Standard Accuracy $\epsilon = 10^{-4}$	Extended Accuracy $\epsilon = 10^{-11}$
default	58	54
default, scaled	54	50
full cycles	60	60
full cycles, scaled	58	57
2x2/1x1	58	58
2x2/1x1, scaled	58	57
2x2/3x3	58	58
2x2/3x3, scaled	58	56

Extended accuracy - time for solve

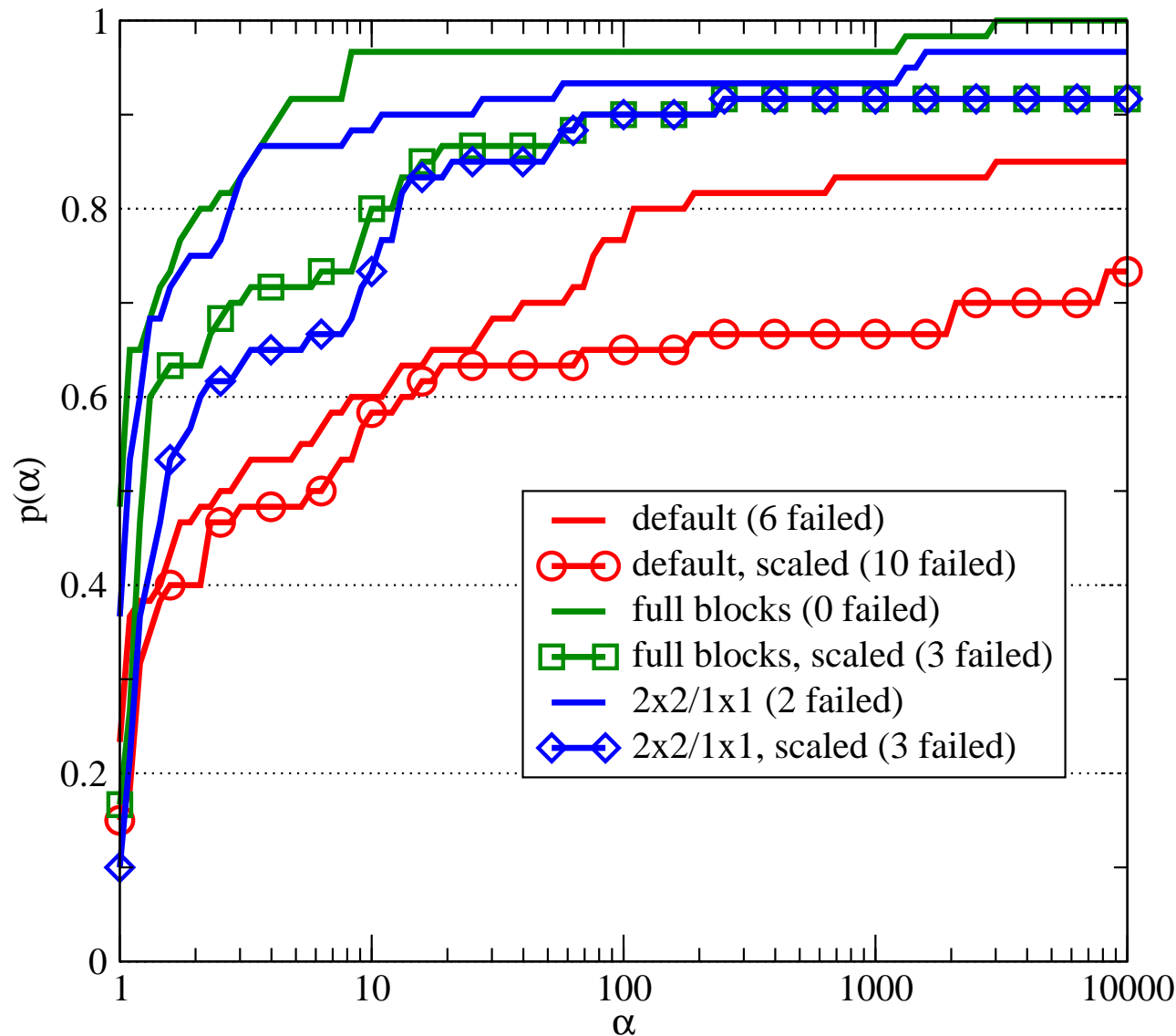


- **Time** for forward- and backward substitution
- Preprocessing strategies only slightly slower than default

Conclusion

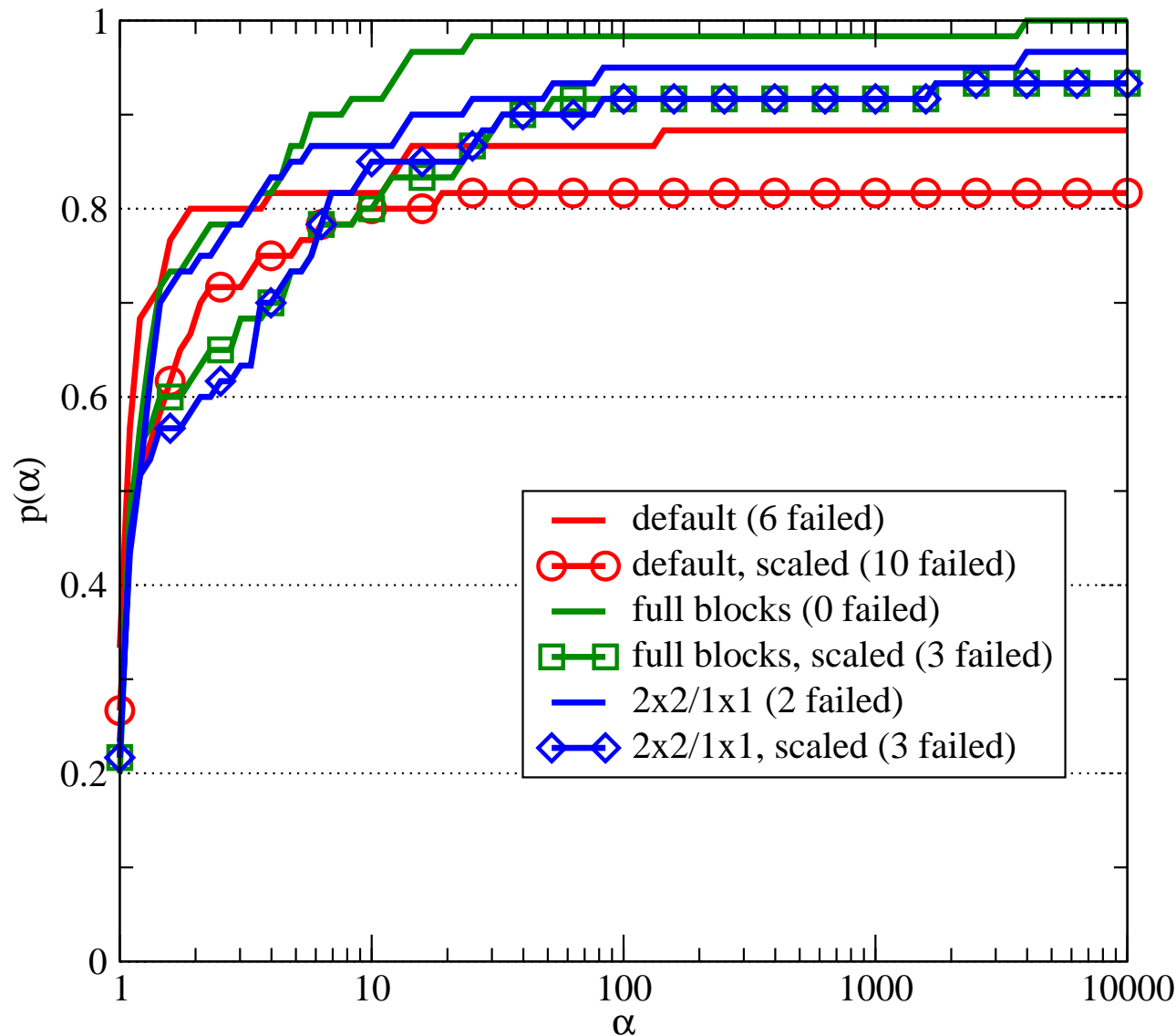
- Investigation and comparison of different preprocessing strategies for PARDISO based on maximum weighted matchings for symmetric indefinite systems
- PARDISO with iterative refinement a good choice in general
- Preprocessing advantageous if iterative refinement is a problem
- Preprocessing strategies **full blocks** and **2x2/1x1** beneficial for accuracy
- Scaling the matrices deteriorates accuracy
- Splitting large blocks is not important and sometimes even disadvantageous

Extended accuracy - first scaled residual



- Scaled residuals without iterative refinement for extended accuracy
- Results similar to standard accuracy
- **Full blocks** strategy best, and **2x2/1x1** a close second

Extended accuracy - second scaled residual



- Scaled residuals after one step iterative refinement for extended accuracy
- Results similar to standard accuracy
- **Full blocks** strategy best, and **2x2/1x1** a close second