Deciding Non-Compressible Blocks in Sparse Direct Solvers using Incomplete Factorization

Esragul Korkmaz, Mathieu Faverge, Grégoire Pichon and Pierre Ramet

Table of contents

K-Way Clustering

Nested Dissection

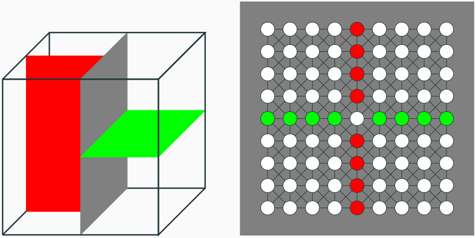


Figure 1: Regular cube with two levels of nested dissection. The right figure shows the first (grey) separator and the second level separator traces (red and green) on it.

- Recursive method
- Generates two balanced parts with minimum size separator

Advantage

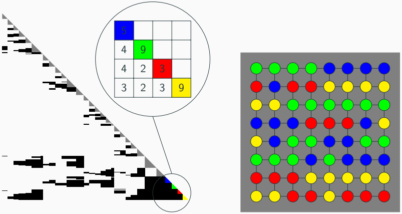
✓ Reduces fill-in and improves parallelism

Discussions: Ordering within separator does not change fill-in

=> Can be improved for granularity

=> Can be improved for compressibility

Reordering



(a) Symbolic factorization of reordering

(b) First separator clusters of reordering

Figure 2: $8 \times 8 \times 8$ Laplacian partitioned using SCOTCH with reordering with smart splitting. Zoomed figure shows the total update counts on the each block. The right figure represents the clustering of the unknowns inside the first separator.

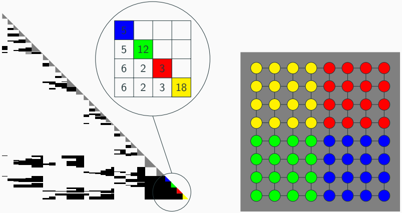
Aim: Improve granularity

✓ Reduced number of updates on blocks

Discussions: Clustering among far nodes of the graph

✗ Low separator compressibility

K-Way Clustering



(a) Symbolic factorization of K-Way

(b) First separator clusters of K-Way

Figure 3: $8 \times 8 \times 8$ Laplacian partitioned using SCOTCH with K-Way clustering. Zoomed figure shows the total update counts on the each block. The right figure represents the clustering of the unknowns inside the first separator.

Aim: Clustering with small diameter and fewer neighbors

✓ High separator compressibility

Discussions: Increased number of updates on blocks

=> Reordering within clusters

- ✓ Still improved granularity
- ✗ Worse granularity than reordering

Non-Compressible Blocks Decision

Block Low-Rank (BLR) Compression Format

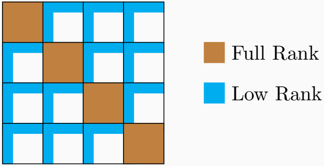


Figure 4: BLR representation of a dense matrix which is clustered into four. Brown color shows the dense diagonal blocks, while the blue color stands for the low-rank representation of the compressed matrices.

- PaStiX sparse solver uses BLR
- Diagonal blocks are dense
- Off-diagonal blocks are compressed through some admissibility criteria
- Two scenario depending on when to compress:
  * Minimal Memory
  * Just in Time

Compression Scenarios

Algorithm 1 Minimal Memory - Just in Time Scenarios

for k=1 to N_{cblk} **do**
    Compress(A_{ij}) // Compress blocks
end for
for k=1 to N_{cblk} **do**
    Factorize
    for each off-diagonal block A_{ij} in *cblk* **do**
        Compress(A_{ij}) // Compress blocks
    end for
    Solve
    Update
end for

- Minimal Memory - Compression before any numerical operations

✓ Reduces memory footprint

✗ Runs in long time

- Just in Time - Panel-wise compression during factorization

✓ Eliminates costly low-rank operations

✓ Reduces time to solution

✗ Uses memory as much as in full rank

Incomplete LU (ILU) Factorization

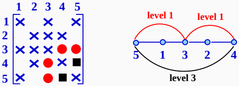


Figure 5: Fill-in levels of the coefficients during elimination.

Algorithm 2 Symbolic ILU(*maxlevel*) Factorization

for a_{ij} in A **do**
    if $a_{ij} \neq 0$ **then**
        $lev(a_{ij}) = 0$
    else
        $lev(a_{ij}) = \infty$
    end if
end for
for k=1 to n-1 **do**
    for i=k+1 to n **do**
        if $lev(a_{ik}) < maxlevel$ **then**
            for j=k+1 to n **do**
                $lev(a_{ij}) = min(lev(a_{ij}), lev(a_{ik}) + lev(a_{kj}) + 1)$
            end for
        end if
    end for
end for

- LU factorization: $A = LU$
- ILU factorization: $A = LU + R$
- As fill-in level increases, the coefficient value gets lower
- Can be performed in block-wise manner
- Preselect blocks with lower levels:
  - Preselected blocks are compressed during factorization
  - All other blocks are compressed in the beginning
- Aim:
  - Preselected blocks do not improve memory footprint much
  - ✗ Little more extra memory usage
  - ✓ Improve time to solution drastically

New ILU(*maxlevel*) Heuristic

Algorithm 3 Minimal Memory - Just in Time - ILU(*maxlevel*) Scenarios

```
for k=1 to N_cblk do
    if level(A_ij) > maxlevel then
        Compress(A_ij) // Compress blocks
    end if
end for
for k=1 to N_cblk do
    Factorize
    for each off-diagonal block A_ij in cblk do
        if level(A_ij) <= maxlevel then
            Compress(A_ij) // Compress blocks
        end if
    end for
    Solve
    Update
end for
```

Experiments

Reordering VS K-Way (Minimal Memory)

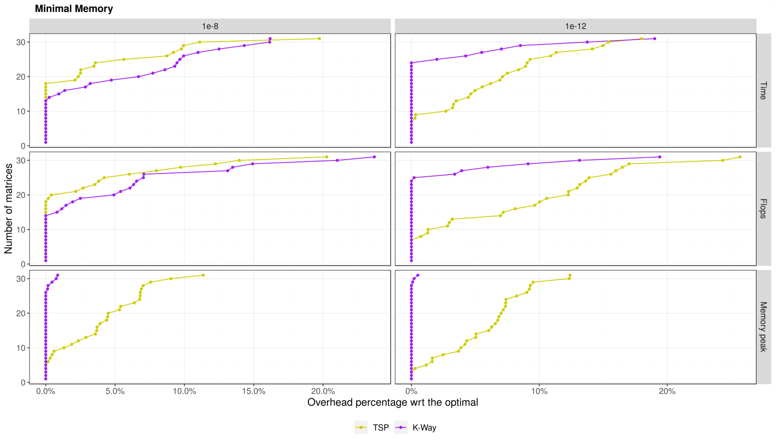


Figure 6: Minimal Memory scenario profiles. The x-axis stands for $\frac{method}{optimal} - 1$. The y-axis stands for 31 real-case matrices.

- K-Way provides:
- ✓ Reduced memory footprint
- ✓ Improved time/flops for high precision

Reordering VS K-Way (Just in Time)

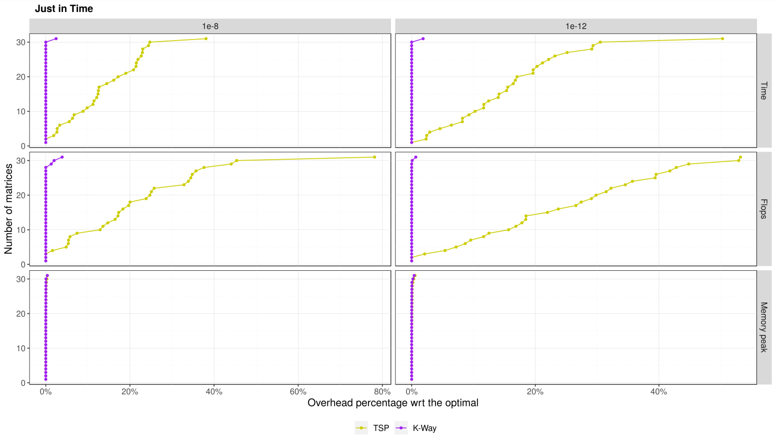


Figure 7: Just in Time scenario profiles. The x-axis stands for $\frac{method}{optimal} - 1$. The y-axis stands for 31 real-case matrices.

- K-Way provides:
  - ✓ Reduced memory footprint
  - ✓ Improved time/flops

Compressibility

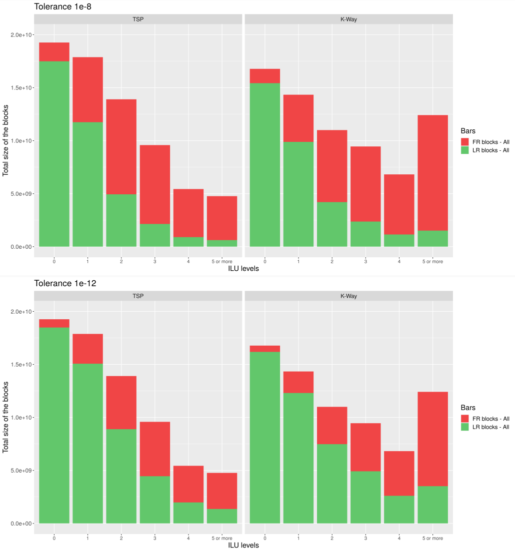


Figure 8: Compressibility figures for 10^{-8} (at the top) and 10^{-12} (at the bottom).

- K-Way reduces compressibility of low levels

=> Better for the new heuristic

- Level 0 is not very compressible for both precisions
- Level 1 is not very compressible for high precisions

=> Adopt ILU(0) for low precision

=> Adopt ILU(1) for high precision

Compressibility

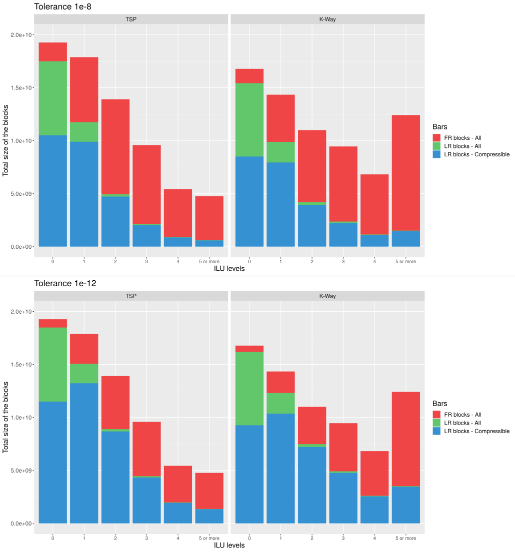


Figure 9: Compressibility figures for 10^{-8} (at the top) and 10^{-12} (at the bottom).

- K-Way reduces compressibility of low levels

=> Better for the new heuristic

- Level 0 is not very compressible for both precisions
- Level 1 is not very compressible for high precisions

=> Adopt ILU(0) for low precision

=> Adopt ILU(1) for high precision

- Expected huge speedup as ratio of preselected blocks is high

ILU(*maxlevel*) Heuristic in Sequential (Reordering vs K-Way)

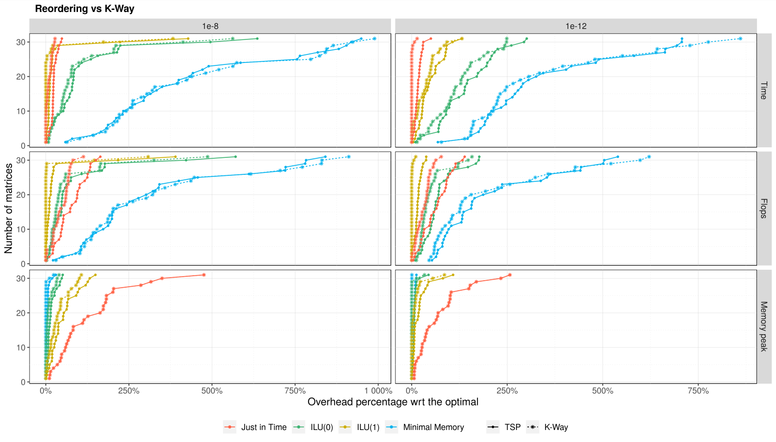


Figure 10: The x-axis stands for $\frac{method}{optimal} - 1$. The y-axis stands for 31 real-case matrices.

✓ K-Way always improves the new heuristic in terms of time, flops and memory usage

ILU(*maxlevel*) Heuristic in Sequential (K-Way)

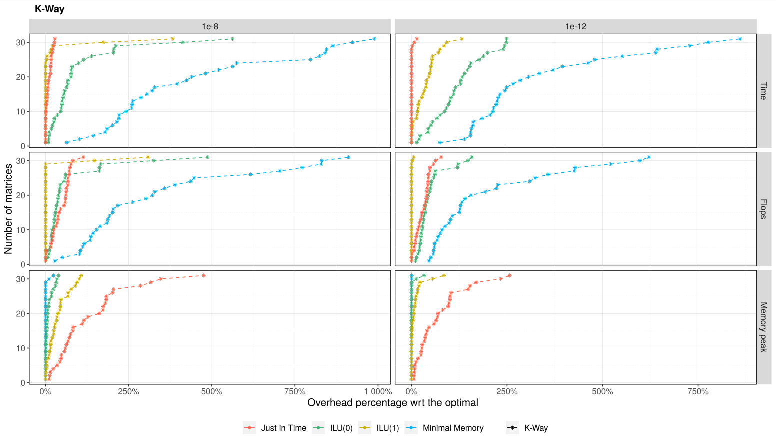


Figure 11: The x-axis stands for $\frac{method}{optimal} - 1$. The y-axis stands for 31 real-case matrices.

✓ Highly improved flops/time with new heuristic as levels increases

✗ Memory usage should be under control:

=> Level 0 is for low precision

=> Level 1 is for high precision

ILU(*maxlevel*) Heuristic in Multithreaded (Reordering vs K-Way)

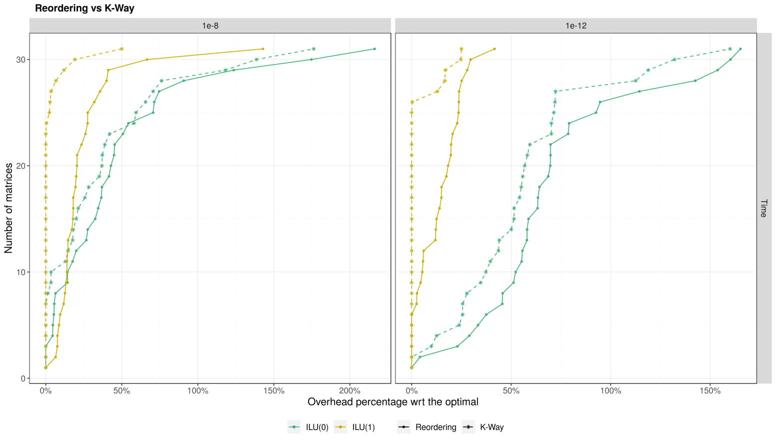


Figure 12: The x-axis stands for $\frac{method}{optimal} - 1$. The y-axis stands for 31 real-case matrices. Experiments used 24 threads.

✓ K-Way improves the new heuristic in terms of time

ILU(*maxlevel*) Heuristic in Multithreaded (K-Way)

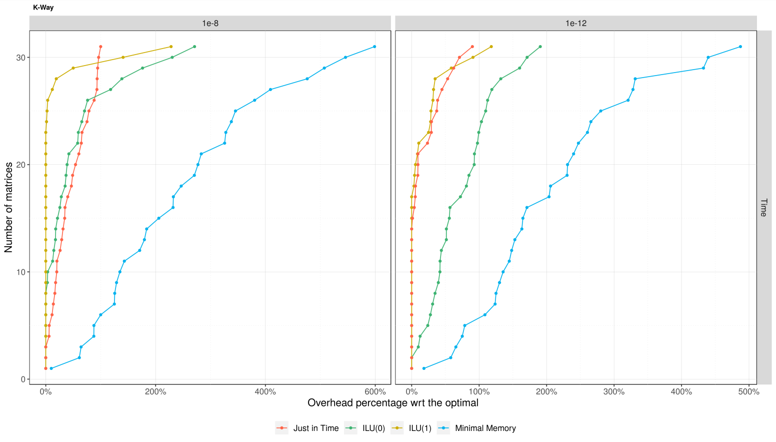


Figure 13: The x-axis stands for $\frac{method}{optimal} - 1$. The y-axis stands for 31 real-case matrices. Experiments used 24 threads.

- Level 0 is for low precision
- Level 1 is for high precision

✓ New heuristic is as fast as Just in Time

Conclusion/Future Work

- K-Way improves all memory footprint, flops and time to solution for Minimal Memory in high precision
- K-Way always improves flops and time to solution for Just in Time
- New heuristic in sequential provides huge speedup and reduces flops with slight memory increase
- New heuristic in multithreaded is even as fast as Just in Time
- New heuristic should be used with K-Way
- Future work:
  * Tuning between ILU(0) and ILU(1) according to precision in PaStiX
  * Further improving the compressibility by aligning the traces in the nested dissection better[1]

[1] Grégoire Pichon. On the use of low-rank arithmetic to reduce the complexity of parallel sparse linear solvers based on direct factorization techniques. PhD Thesis. Université de Bordeaux, 2018.

Thank You!

ILU(*maxlevel*) Heuristic in Sequential (Reordering)

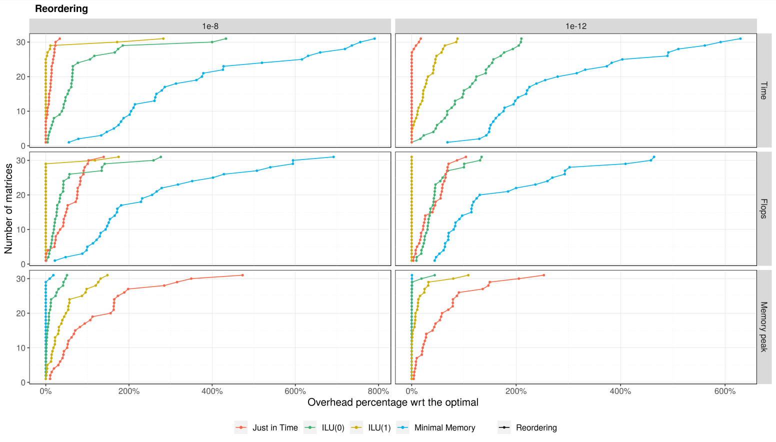


Figure 14: The x-axis stands for $\frac{method}{optimal} - 1$. The y-axis stands for 31 real-case matrices.

✓ Highly improved flops/time with new heuristic as levels increases

✕ Memory usage should be under control:

- Level 0 is for low precision
- Level 1 is for high precision

All Sequential Results Together

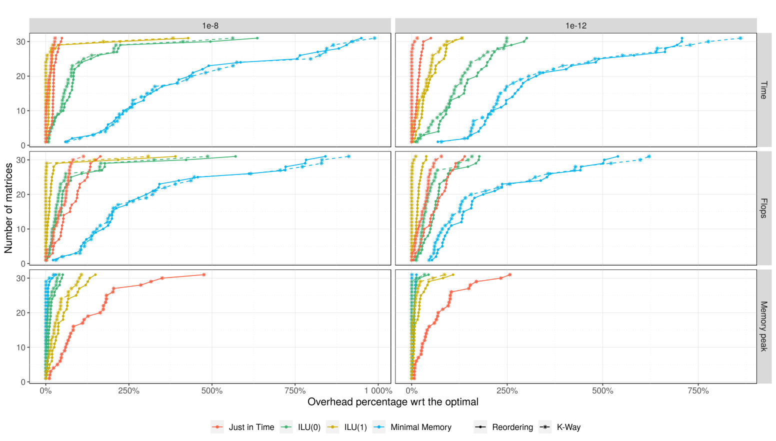


Figure 15: The x-axis stands for $\frac{method}{optimal} - 1$. The y-axis stands for 31 real-case matrices.

All Multithreaded Results Together

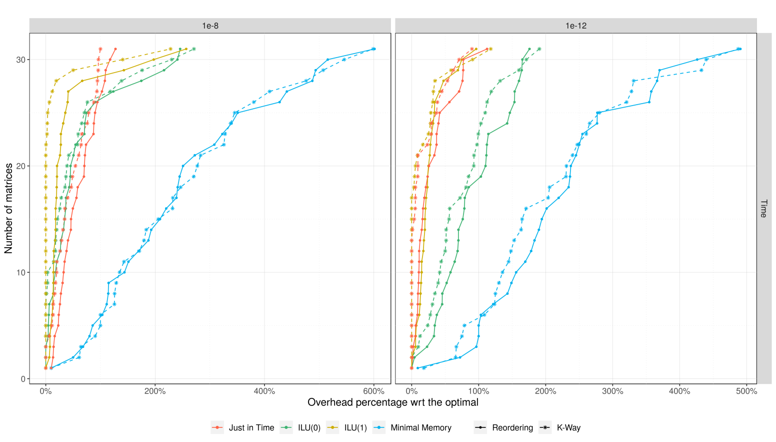


Figure 16: The x-axis stands for $\frac{method}{optimal} - 1$. The y-axis stands for 31 real-case matrices. Experiments used 24 threads.