Recent developments in HiPO

Filippo Zanetti

University of Edinburgh, School of Mathematics

HiGHS workshop, Edinburgh, 1 June 2026

Timeline

- beta ⟵ HiPO available on a dedicated branch
- 1.12 ⟵ HiPO is officially released
- 1.13
- 1.14
- 1.15 ⟵ HiPO available on `highspy`

Results

Time limit: 5000 seconds
Tolerance: 10^{-8}
Crossover: `off` (the meaning of `choose` has changed and cannot be compared)

Collection	Type	#	Solved		Need IPX		Time[1]	
			1.12	1.15	1.12	1.15	1.12	1.15
Netlib	LP	98						
Mittelmann	LP	49						
PyPSA	LP	45						
Maros-Meszaros	QP	138						

[1]Shifted geometric mean with shift of 10s
[2]Using active-set solver

Results

Time limit: 5000 seconds
Tolerance: 10^{-8}
Crossover: `off` (the meaning of `choose` has changed and cannot be compared)

Collection	Type	#	Solved		Need IPX		Time[1]	
			1.12	1.15	1.12	1.15	1.12	1.15
Netlib	LP	98	95		8		2.6	
Mittelmann	LP	49	35		22		465.0	
PyPSA	LP	45	45		1		44.6	
Maros-Meszaros	QP	138	104[2]		-		44.0[2]	

[1] Shifted geometric mean with shift of 10s
[2] Using active-set solver

Timeline

- beta
  - **Improvement of memory allocation for factorisation**
  - **Improvement of normal equations matrix construction**
- 1.12
- 1.13
- 1.14
- 1.15

Memory allocation

The memory required to store the factorisation can be allocated once and then reused.

Problem	Time for allocations	
	before	after
`fome13`	3.1	2.1
`neos`	15.0	10.5
`set-cover`	4.9	0.9

On average, 5% reduction in the time used for memory allocations.

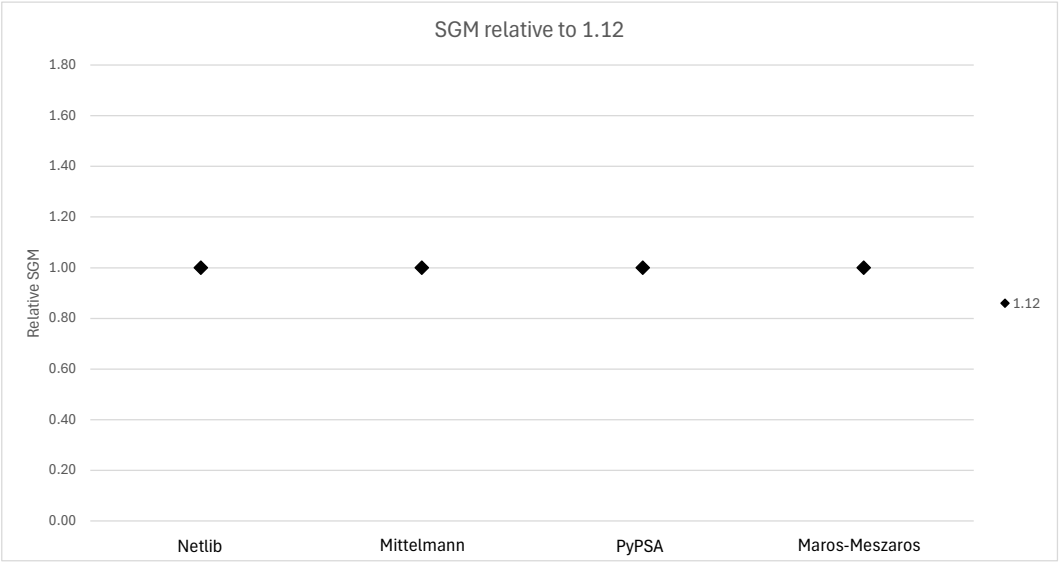
Normal equations

Building the normal equations structure (only once) and values (at each iteration) has been made more efficient.

Problem	NE structure time		NE values time	
	before	after	before	after
`dlr1`	32.7	0.2	0.11	0.03
`fhnw-bin1`	89.4	1.5	0.69	0.18
`tpl-tub-ws`	15.5	0.1	0.16	0.03

On average, 75% reduction in the time to build the structure and 25% reduction in the time to build the values.

SGM evolution



Timeline

- beta
- 1.12
  - **Better iterative refinement**
  - **Stack-based management of the elimination tree**
  - **64-bit integers inside of the factorisation**
  - **Custom Metis and AMD**
- 1.13
- 1.14
- 1.15

Iterative refinement

Iterative refinement is computed on the full 6×6 linear system

$$\begin{bmatrix} A & & & & & \\ I & & -I & & & \\ I & & & I & & \\ & A^T & & & I & -I \\ & & Z^l & & X^l & \\ & & & Z^u & & X^u \end{bmatrix} \begin{bmatrix} \Delta x \\ \Delta y \\ \Delta x^l \\ \Delta x^u \\ \Delta z^l \\ \Delta z^u \end{bmatrix} = \begin{bmatrix} b - Ax \\ l - x + x^l \\ u - x - x^u \\ c - A^T y - z^l + z^u \\ \sigma\mu e - X^l Z^l e \\ \sigma\mu e - X^u Z^u e \end{bmatrix}.$$

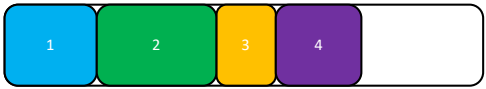
This improves robustness overall.

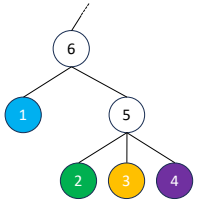
Stack-based memory allocation for elimination tree

If the elimination tree is processes in serial, the memory required by the Schur complements can be managed as a **stack**.

This reduces the number of memory allocations and deallocations.

Making this work when the tree is processed in parallel is still work-in-progress.





64-bit integers

The following error would sometimes occur:

```
> Factorisation statistics
> Size:                          6.03e+06
> Nnz:                           7.17e+09
> Fill-in:                       227.24
> Serial memory:                 6.2e+01 GB
> Flops:                         1.4e+14
>
> ERROR:   Both NE and AS failed analyse phase
```

This happens because the factorisation requires $7 \cdot 10^9$ entries, but 32-bit integers can only fit around $2 \cdot 10^9$.

Using **64-bit integers** to store information about the nonzero entries of the factorisation avoids this issue.

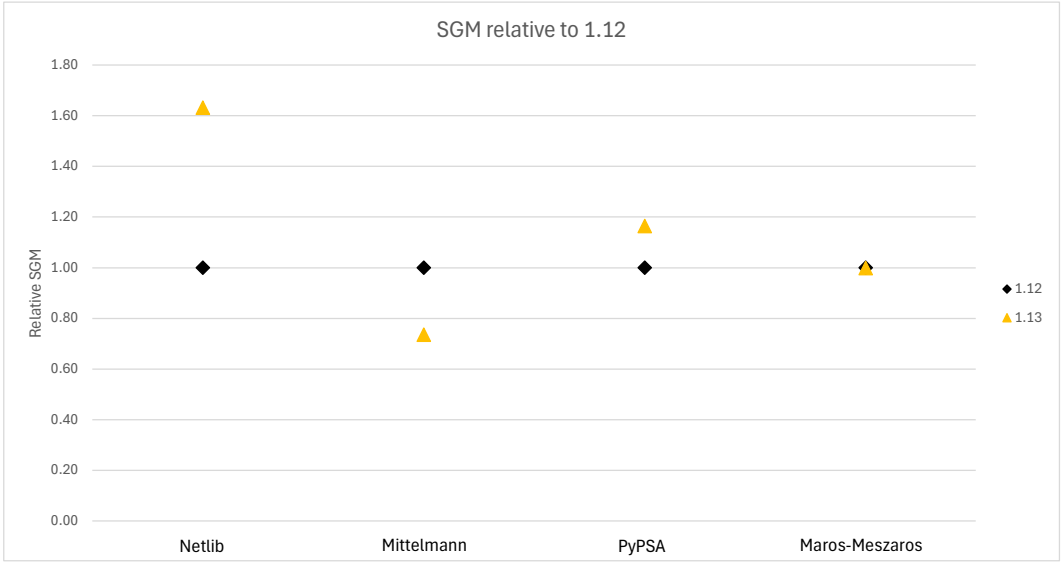
The original problem still needs to fit in 32-bit integers, unless `HIGHSINT64` is used.

Metis and AMD

The source code of **Metis** (nested dissection) and **AMD** (approximate minimum degree) has been included as part of HiGHS, with minor changes.
A simple heuristic is used to choose between Metis and AMD orderings.

Problem	Ordering time		Factorisation time	
	metis	amd	metis	amd
`cont1`	1.2	0.1	2.2	0.3
`fhnw-bin1`	41.2	97.5	8.4	36.2
`pds-100`	1.1	0.6	0.7	4.2

SGM evolution



Timeline

- beta
- 1.12
- 1.13
  - **Extension to QPs**
  - **Improved termination criterion**
  - **Better supernode amalgamation**
  - **Orderings and analyse phase in parallel**
  - **Better scaling**
- 1.14
- 1.15

HiPO for QPs

Extending HiPO to solve **convex QPs** required few changes. Some features:

- The exact same algorithm and factorisation is used as for LPs.
- The normal equations can be used if the quadratic term is separable (diagonal).
- QP presolve is not available in HiGHS, but HiPO performs some simple pre-processing.
- QP crossover is not available.

For large problems, HiPO for QPs can be vastly superior to the current active-set method.

Termination

The HiPO **termination criterion** now has a feedback loop with the HiGHS solution quality check:

- Before: HiPO can terminate with status `optimal`, but HiGHS can consider the solution as not optimal.
- Now: if HiPO considers the solution optimal, it requires a check from HiGHS `kktCheck`. If the test does not pass, more iterations are performed.

As a consequence, the behaviour of HiPO for different `run_crossover` options has changed and it is now **strongly recommended to use `run_crossover=choose` when crossover is not required**, rather than `run_crossover=off`.

Supernodes amalgamation

A new **supernode amalgamation** strategy has been introduced that considerably reduces the time for the factorisation for some problems. This can be improved further with more work.

Problem	Factorisation time	
	before	after
`a2864`	41.2	22.3
`graph40-40`	1.4	0.6

Parallel analyse phase

The following tasks are executed in parallel during the analyse phase:

- Building the normal equations structure,
- Running Metis on the augmented system,
- Running AMD on the augmented system.

Once they terminate, if the normal equations is not too dense, the following tasks are executed in parallel:

- Running Metis on the normal equations,
- Running AMD on the normal equations.

It brings 5-15% savings in total runtime, while still being able to select the better ordering between Metis and AMD.

Scaling

Scaling the problem before solving it is fundamental for the robustness of an IPM.

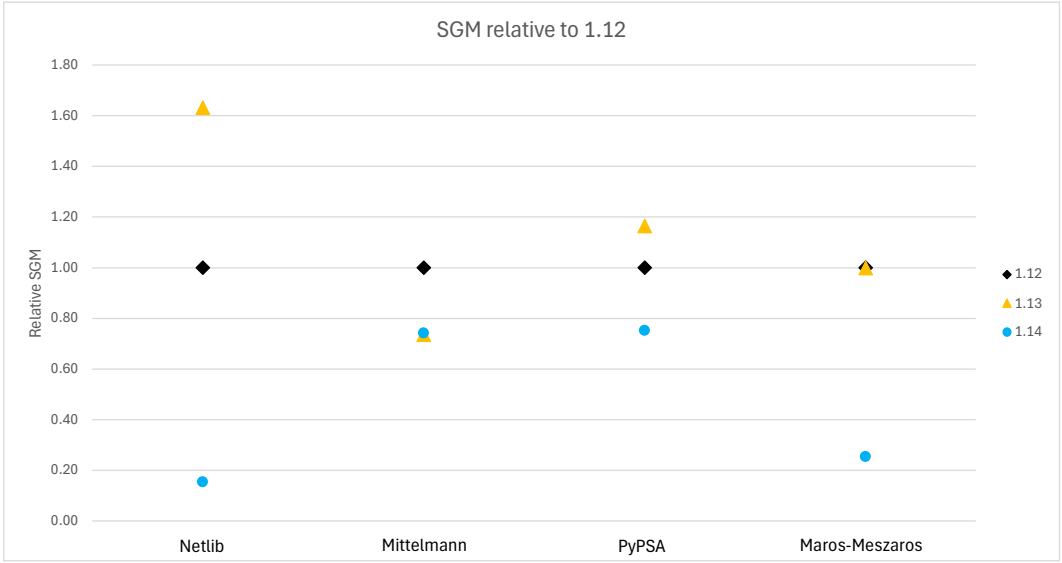
HiPO was using the **Curtis-Reid** scaling, which is found by solving

$$\min_{R,C} \sum_{(i,j)\,|\,A_{ij}\neq 0} \log^2(R_i \cdot |A_{ij}| \cdot C_j).$$

It now uses **∞-norm equilibration**, i.e., it aims at obtaining a scaled matrix A with all rows and all columns having ∞-norm equal to 1.

The new scaling improves **robustness** and significantly reduces the number of iterations needed for some problems.

SGM evolution



Timeline

- beta
- 1.12
- 1.13
- 1.14
  - **Better strategy for free variables**
  - **Faster solves for small supernodes**
- 1.15

Free variables

Free variables can be problematic for interior point methods.

They are now reformulated using an artificial upper and lower bound.

These bounds may be increased or decreased during the iterations, if the variable gets too close to them.

This improves robustness for problems with many free variables.

Solves for small supernodes

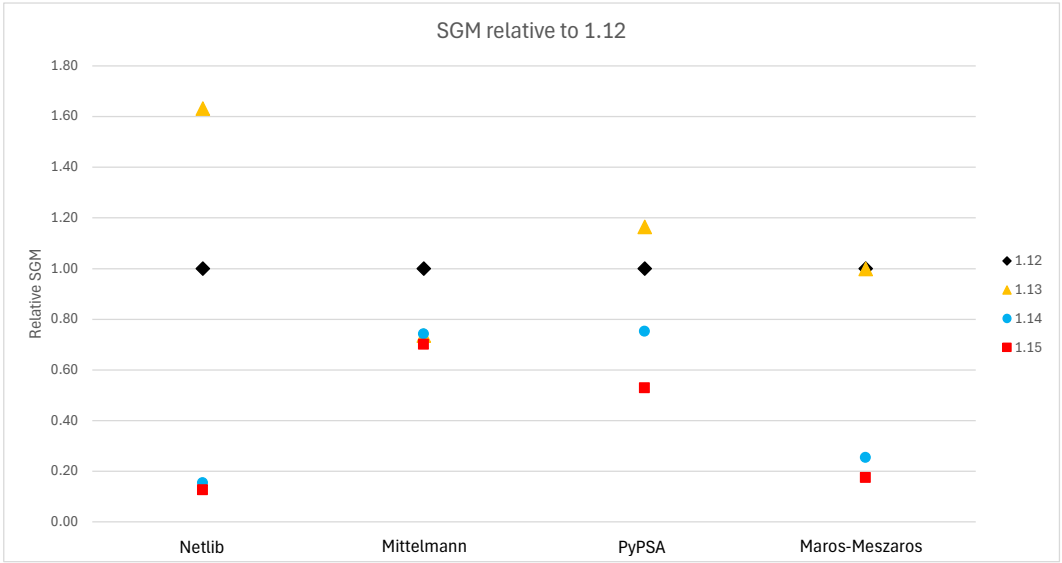
For some very sparse problems, performing triangular solves with the factorisation can be a bottleneck.

If a supernode is small, it is better to avoid using BLAS and perform the operations explicitly instead.

This can bring up to 20-30% improvement in runtime for sparse problems.

Problem	Runtime	
	before	after
`degme`	422.3	362.3
`neos`	269.0	175.1
`trex-2-1h`	114.1	82.5

SGM evolution



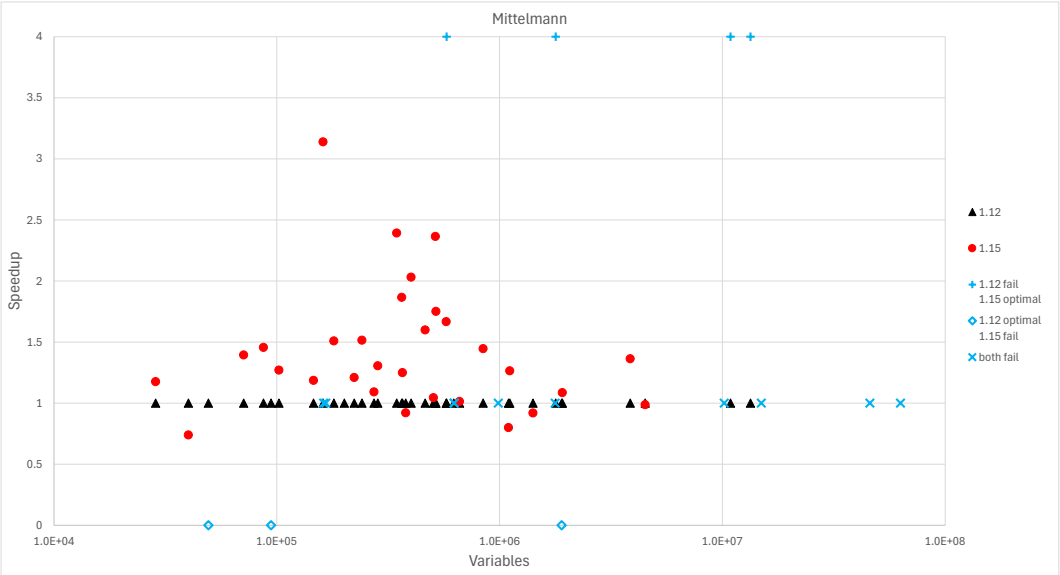
Results

Time limit: 5000 seconds
Tolerance: 10^{-8}
Crossover: `off` (the meaning of `choose` has changed and cannot be compared)

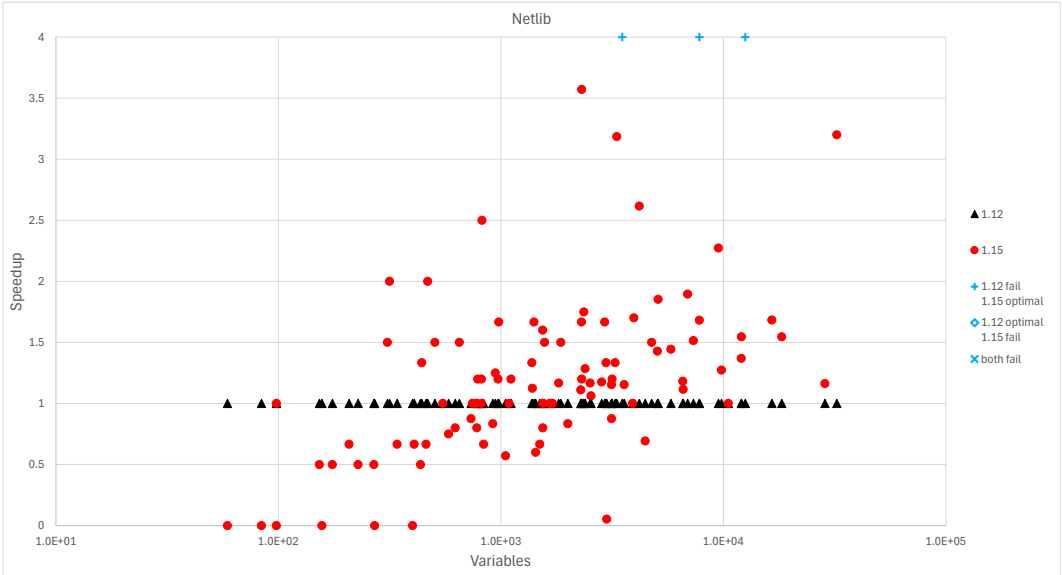
Collection	Type	#	Solved		Need IPX		Time[1]	
			1.12	1.15	1.12	1.15	1.12	1.15
Netlib	LP	98	95	98	8	0	2.6	**0.3**
Mittelmann	LP	49	35	36	22	7	465.0	**325.7**
PyPSA	LP	45	45	45	1	0	44.6	**23.6**
Maros-Meszaros	QP	138	104[2]	126	-	-	44.0[2]	**7.6**

[1] Shifted geometric mean with shift of 10s
[2] Using active-set solver

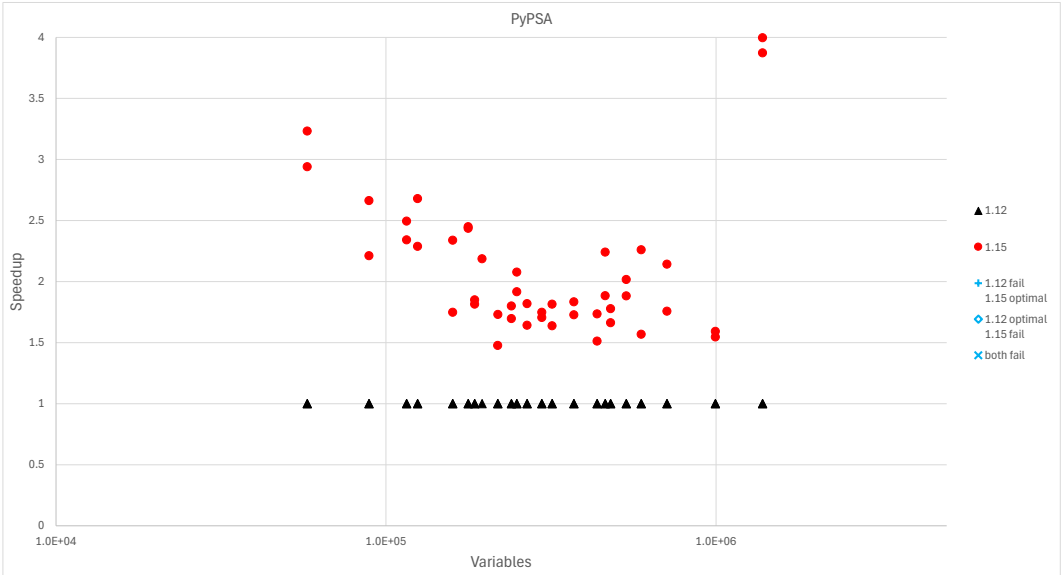
Results


Mittelmann
Speedup
Variables
4
3.5
3
2.5
2
1.5
1
0.5
0
1.0E+04
1.0E+05
1.0E+06
1.0E+07
1.0E+08
1.12
1.15
1.12 fail
1.15 optimal
1.12 optimal
1.15 fail
both fail

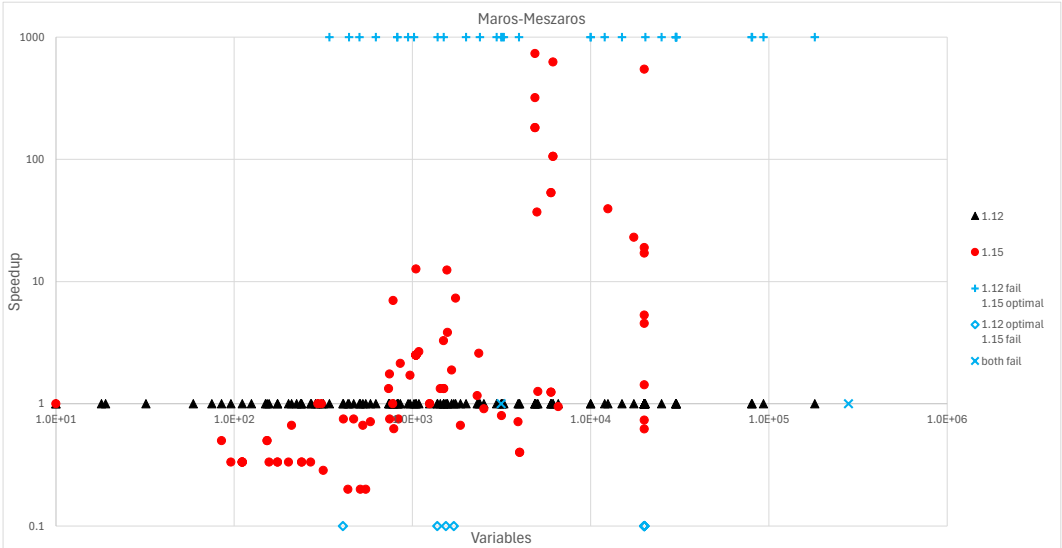
Results


Netlib
Speedup
Variables
4
3.5
3
2.5
2
1.5
1
0.5
0
1.0E+01
1.0E+02
1.0E+03
1.0E+04
1.0E+05
1.12
1.15
1.12 fail
1.15 optimal
1.12 optimal
1.15 fail
both fail

Results


PyPSA
Speedup
Variables
4
3.5
3
2.5
2
1.5
1
0.5
0
1.0E+04
1.0E+05
1.0E+06
1.12
1.15
1.12 fail
1.15 optimal
1.12 optimal
1.15 fail
both fail

Results


Maros-Meszaros
Speedup
1000
100
10
1
0.1
1.0E+01
1.0E+02
1.0E+03
1.0E+04
1.0E+05
1.0E+06
Variables
1.12
1.15
1.12 fail
1.15 optimal
1.12 optimal
1.15 fail
both fail

Timeline

- beta
- 1.12
- 1.13
- 1.14
- 1.15
  - **Future developments**

Future developments

- Stack memory management in parallel
- Further improvements to supernode amalgamation
- Up-looking factorisation
- Folding?