Engineered algorithms in Linear Algebra [I]

David Saunders and Zhendong Wan- U of Delaware

November 12, 2004

Abstract:

Engineered algorithms are attempts to provide, from a range of algorithm choices, the best method for the given input. The adjective "engineered" is meant to suggest that the choice may be a hybrid of several algorithms and that a healthy mix of asymptotic complexity information and consideration of problem instances previously encountered in practice will be used in determining the structure of the hybrid algorithm. We will discuss two engineered algorithms in detail, an adaptive matrix rank algorithm and a hybrid Smith normal form. Some discussion of the general issues of engineered algorithm design may follow.

[I]Supported by NSF grants 0098284 and 0112807

Outline

- Adaptive rank algorithm
- Engineered Smith form algorithm
  - definition and applications
  - Algorithm history
  - New adaptive algorithm
  - Some experimental results
- Summary

Rank of sparse integer matrix

- With high probability, integer rank = rank mod random wordsized prime.
- Blackbox (BB) methods are excellent for large sparse matrices over finite fields. Wiedemann, Kaltofen-Saunders, Turner, Chen et al...
- Sparse elimination (such as SuperLU of Demmel, et al) is excellent on matrices which are small, or slow to fill in. Duran adapted it to work over finite fields.
- Other eliminations are fast by using floating point BLAS.

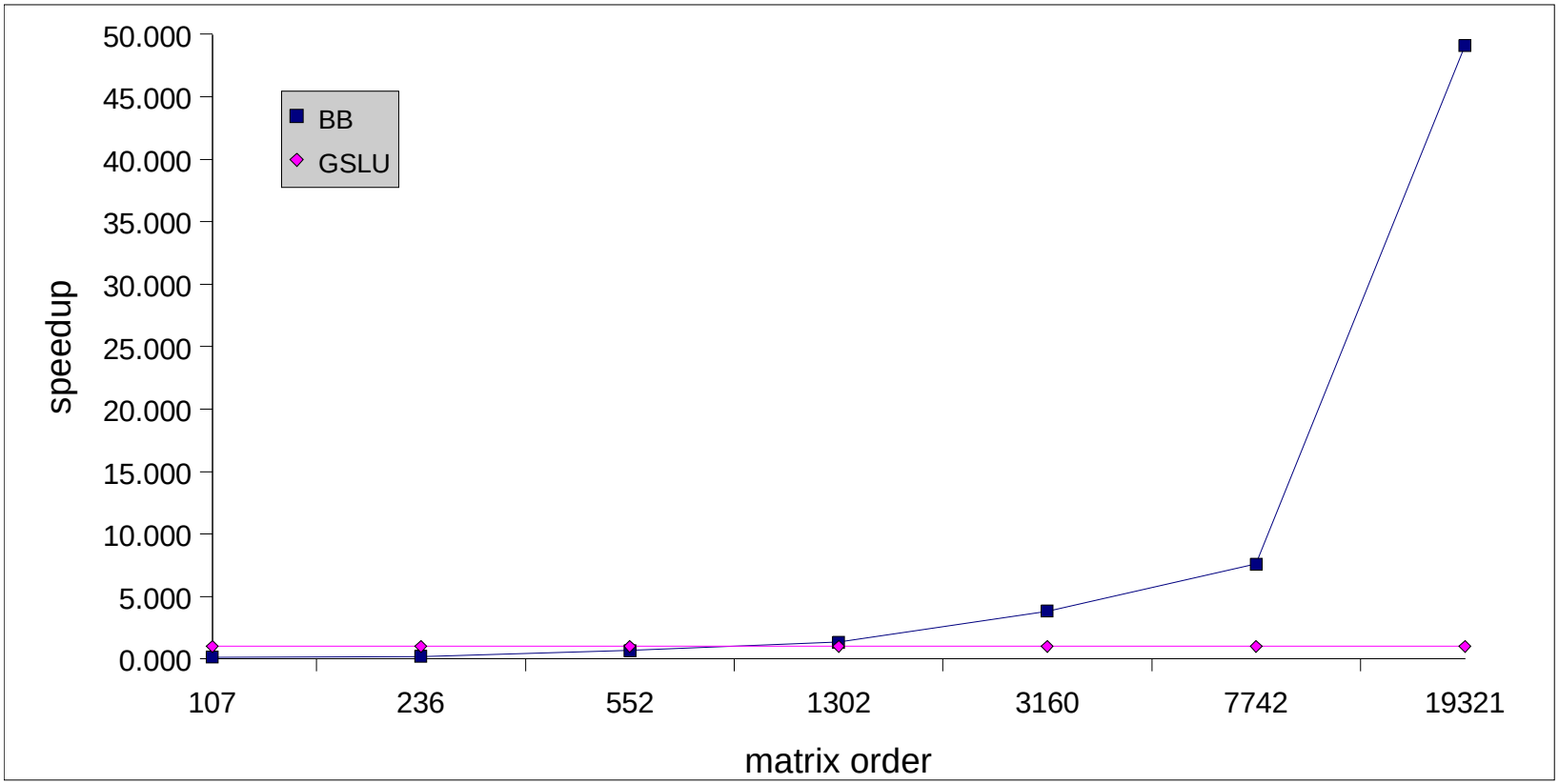
On dense $n \times n$ matrix of rank r:

- BB requires about $4rn^2$ arithmetic ops.
- GSLU requires about $1/3rn^2$ arithmetic ops.
- Which to use? elim (up to memory limitation).
- Measured BB/GSLU ratio is circa 9.5 not 12, due to fast dot product over finite fields.

Sparse Integer Matrices On $n \times n$ matrix with E non-zeroes $(n \leq E \leq n^2)$ and rank r:

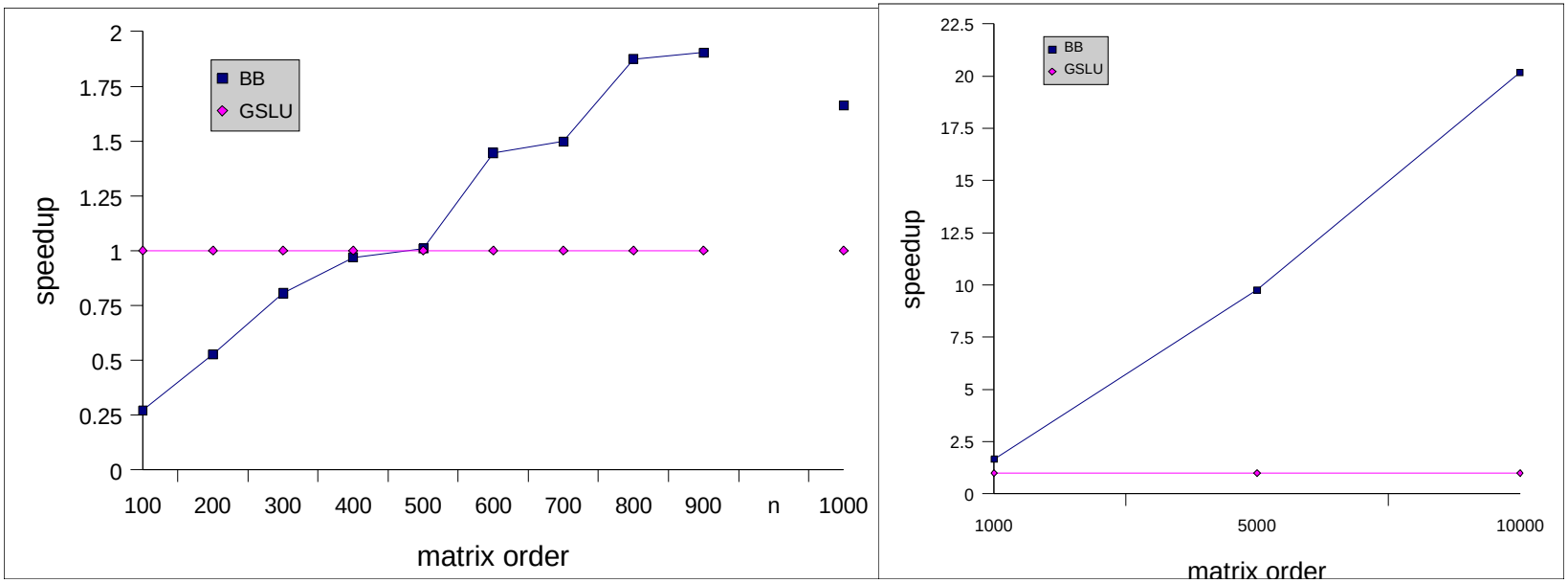
- BB requires about $4rE$ arithmetic ops.
- GSLU requires anywhere from about n to about $1/3rn^2$ ops.
- Which to use???

TF family



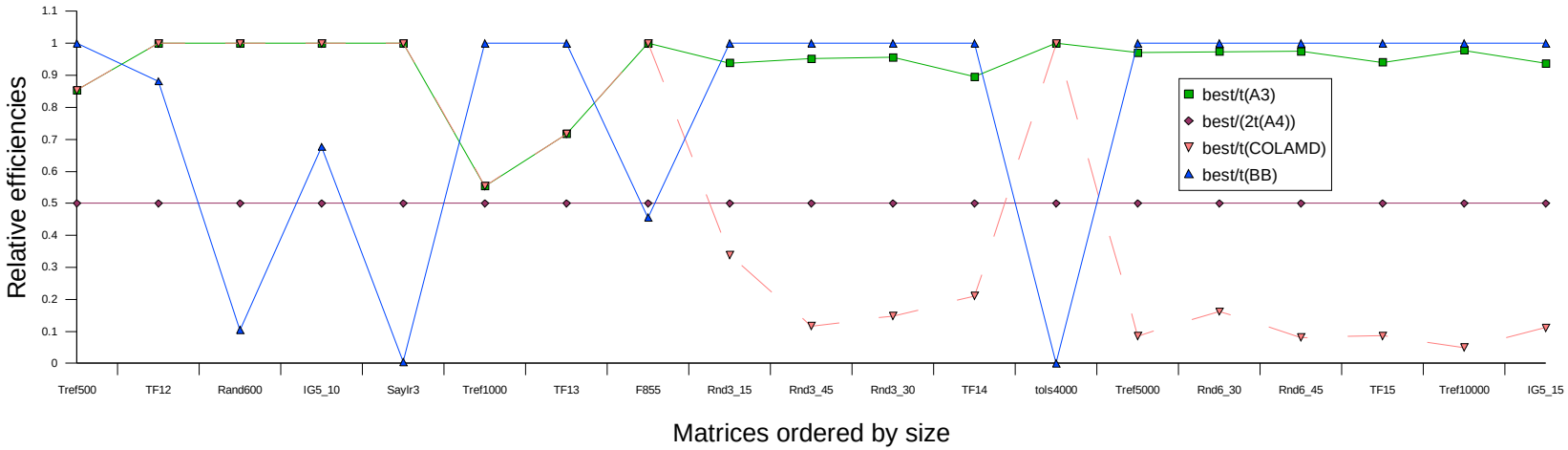
The crossover is near order 1000

Trefethen family



The crossover is near order 500

Adaptive algorithm for rank



- Blackbox method
- Generalized SuperLU
- racing - guaranteed 1/2 efficiency of best of BB, GSLU
- hybrid - elim until BB estimate is faster

An introspective rank algorithm for sparse matrices

- Sparse Elimination
  - good for smaller matrices.
  - for large sparse matrices as long as not too much fill-in.
  - Run time is highly variable, depending on fill-in. In left looking forms, time is an increasing function of step number.
- Blackbox (Wiedemann' or Lanczos')
  - predictable cost
- Adaptive algorithm:

  Run a sparse (left-looking) elimination (such as SuperLU adapted to finite fields) until fill-in slows it down too much. In that case switch to a blackbox method.

In all cases seen so far, the introspective rank algorithm is faster than (timeshared) racing the sparse elimination and blackbox algorithms. In most cases run time is within a few percentage points of the faster of the two.

Multi-algorithms are familiar to CAS designers, but not with runtime switching based on performance estimation. (some STL algs are adaptive based on memory availability or introspective based on recursion depth).

We believe adaptive and introspective methods will be increasingly important and useful.

Smith Form Definition

Definition: For every integer matrix A, there exist unimodular matrices (determinant $= +1, -1$) U and V, and a unique diagonal matrix

$$S = \mathrm{diag}(s_1, s_2, s_3, \ldots)$$

with non-negative diagonal entries s_i, for $i = 1, \cdots, n.$ such that

$$A = USV,$$

and

$$s_i | s_{i+1}.$$

The matrix S is called the Smith Normal Form of A. These s_i are called invariant factors of A.

Also $s_i = d_i/d_{i-1}$, where $d_i = i$-th determinantal divisor = gcd of all $i \times i$ minors.

Applications of the Smith form

1. Structure of finitely generated commutative groups
2. Homology groups (of simplicial complexes, for instance).
   Homology identifies coarse topological properties of an object.
3. Matrix equivalence
4. Linear Diophantine system solving, solvability

Further useful properties.

- Smith form is also defined and unique over $\boldsymbol{Z}_m$
- $S(A) = S_m(A)$, if largest-invariant-factor$(A)|m$.
- $S_{ab}(A) = S_a(A)S_b(A)$, if $\gcd(a, b) = 1$.
- $S_{p^e}(A)$ is particularly easy to compute.
- Cauchy-Binet formula: Let $C_i(A)$ denote the i-th compound matrix of A. Then $C_i(AB) = C_i(A)C_i(B)$.

$$\begin{array}{ccc} & \overset{C_i}{} & \\ A, B & \longrightarrow & C_i(A), C_i(B) \\ \text{mul} \quad \downarrow & & \downarrow \\ AB & \longrightarrow & C_i(AB) \end{array}$$

For example the adjoint is the $n-1$ compound.

EEA: Given integers a, b, get integers g, s, t such that $\gcd(a, b) = g = sa + tb$. Hence,

$$\begin{bmatrix} s & t \\ \frac{-b}{g} & \frac{a}{g} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} g \\ 0 \end{bmatrix},$$

and det is 1.

Example: Given -42 and 68, EEA gives 2, 13, 8, and

$$\begin{array}{ccccc} \text{row op} & \times & A & = & A' \\ \begin{bmatrix} 13 & 8 & . \\ 34 & 21 & . \\ . & . & 1 \end{bmatrix} & \times & \begin{bmatrix} -42 & * & 50 \\ 68 & * & 51 \\ * & * & * \end{bmatrix} & = & \begin{bmatrix} 2 & * & 1058 \\ 0 & * & 2771 \\ * & * & * \end{bmatrix} \end{array}$$

Example

$$A = \begin{bmatrix} -66 & -65 & 20 & -90 & 30 \\ 55 & 5 & -7 & -21 & 62 \\ 68 & 66 & 16 & -56 & -79 \\ 26 & -36 & -34 & -8 & -71 \\ 13 & -41 & -62 & -50 & 28 \end{bmatrix}.$$

$$\text{Smith Form } S = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 4820471082 \end{bmatrix}.$$

$$U = \begin{bmatrix} 0 & -4 & 0 & 0 & 17 \\ -1 & -2155799 & 0 & 8363629 & -7606575 \\ 0 & -27103 & 12205 & 30735 & -10645 \\ -97332885832370 & 3161491831018314 & -1535603236941634 & -3480736996093707 & 1124166218424908 \\ 194665772230104 & -6322983680400346 & 3071206482802913 & 6961474012405487 & -2248332443379604 \end{bmatrix}$$

$$V = \begin{bmatrix} 1 & 717 & -145074077103 & -614119807632672606671239 7 & -1480173382489511933491410080177 9652 \\ 0 & 1 & -202334837 & -8565129874140958443401 & -2064398037597948573029046099 2360 \\ 0 & 0 & 1 & 42331900198616 & 1020298500804459406488 91 \\ 0 & 0 & 0 & -246039914 & -593014143505104075 \\ 0 & 0 & 0 & -1 & -2410235534 \end{bmatrix}.$$

Example: Given -42 and 68, EEA gives 2, 13, 8, and

$$\begin{matrix} \text{row op} & \times & A & = & A' \end{matrix}$$

$$\begin{bmatrix} 13 & 8 & . \\ 34 & 21 & . \\ . & . & 1 \end{bmatrix} \times \begin{bmatrix} -42 & * & 50 \\ 68 & * & 51 \\ * & * & * \end{bmatrix} = \begin{bmatrix} 2 & * & 1058 \\ 0 & * & 2771 \\ * & * & * \end{bmatrix}$$

Suppose the largest invariant factor is 100, so mod 200:

$$\begin{bmatrix} 13 & 8 & 0 \\ 34 & 21 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} -42 & * & 50 \\ 68 & * & 51 \\ * & * & * \end{bmatrix} = \begin{bmatrix} 2 & * & 58 \\ 0 & * & 171 \\ * & * & * \end{bmatrix}$$

Local at p = 5, so mod 125 (knowing lif = 100):

$$\begin{bmatrix} -3 & 0 & 0 \\ -46 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} -42 & * & 50 \\ 68 & * & 51 \\ * & * & * \end{bmatrix} = \begin{bmatrix} 1 & * & -25 \\ 0 & * & 1 \\ * & * & * \end{bmatrix}$$

Smith Form History

$n \times n$ dense matrices with constant size entries, using standard Mat Mul.

Author(s)	Year	Citation	Time Cost	Method
Smith	1861	[7]	UNK	determ
Kannan and Bachem	1979	[6]	Polynomial	D
Iliopoulos	1989	[4]	$O^{\sim}(n^5)$	D, mod det
Hafner and McCurley	1991	[3]	$O^{\sim}(n^5)$	D, mod det
Giesbrecht	1995	[2]	$O^{\sim}(n^4)$	P, mod det
Storjohann	1996	[8]	$O^{\sim}(n^4)$	D, mod det
Eberly, Giesbrecht and Villard	2000	[1]	$O^{\sim}(n^{3.5})$	P, mod s_n
Kaltofen and Villard	2003	[5]	$O^{\sim}(n^{3.33})$	P, mod D_n
Local at a prime p		Folklore	$O^{\sim}(n^4)$	D, mod p^e

Also mention Pan 88, Abbot Bronstein Mulders 1999, Mulders and Storjohann 1999, effective use of last invariant factor, bisection, valence alg.

Smooth / Rough part

we call a number k-*smooth* if it is a product of primes smaller than k and k-*rough* if it is a product of primes no less than k.

$$840225648742796300569804800 = 2^{20} \cdot 3^9 \cdot 5^2 \cdot 7^{18}$$

is 10-smooth, and

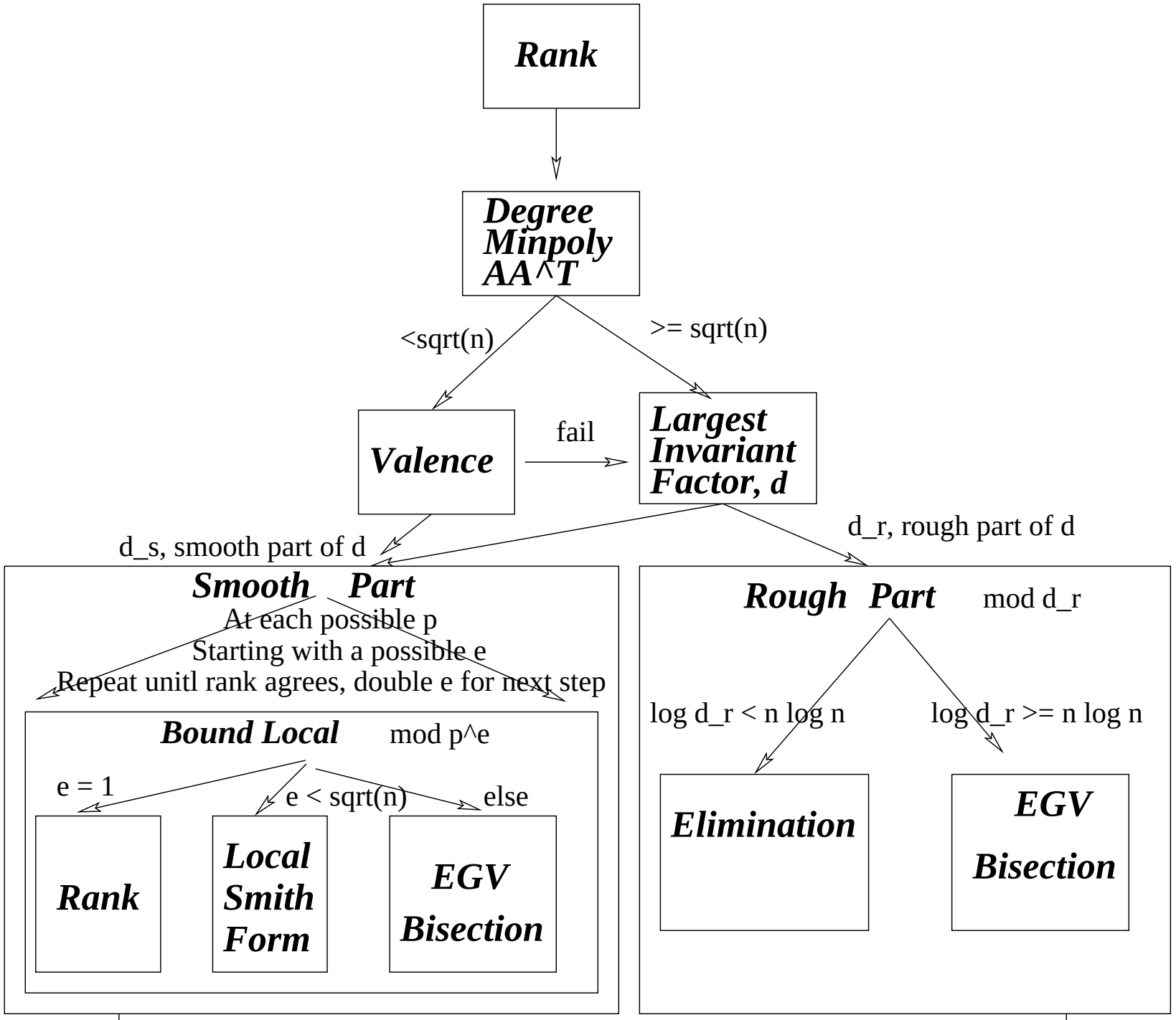
$$847831897 = 11 \cdot \text{ithprime}(1000) \cdot \text{ithprime}(1200)$$

is 10-rough.

Handling the smooth part

- For wordsize moduli Iliopoulos' elimination time is about equivalent in cost to 3 Local eliminations.
- Local elimination mod 2^{32} is about 10 times faster than any other local computation.
- If smooth prime p does not occur in valence or largest invariant factor, just need a fast rank computation mod p.

An engineered, adaptive, hybrid Smith form multi-algorithm

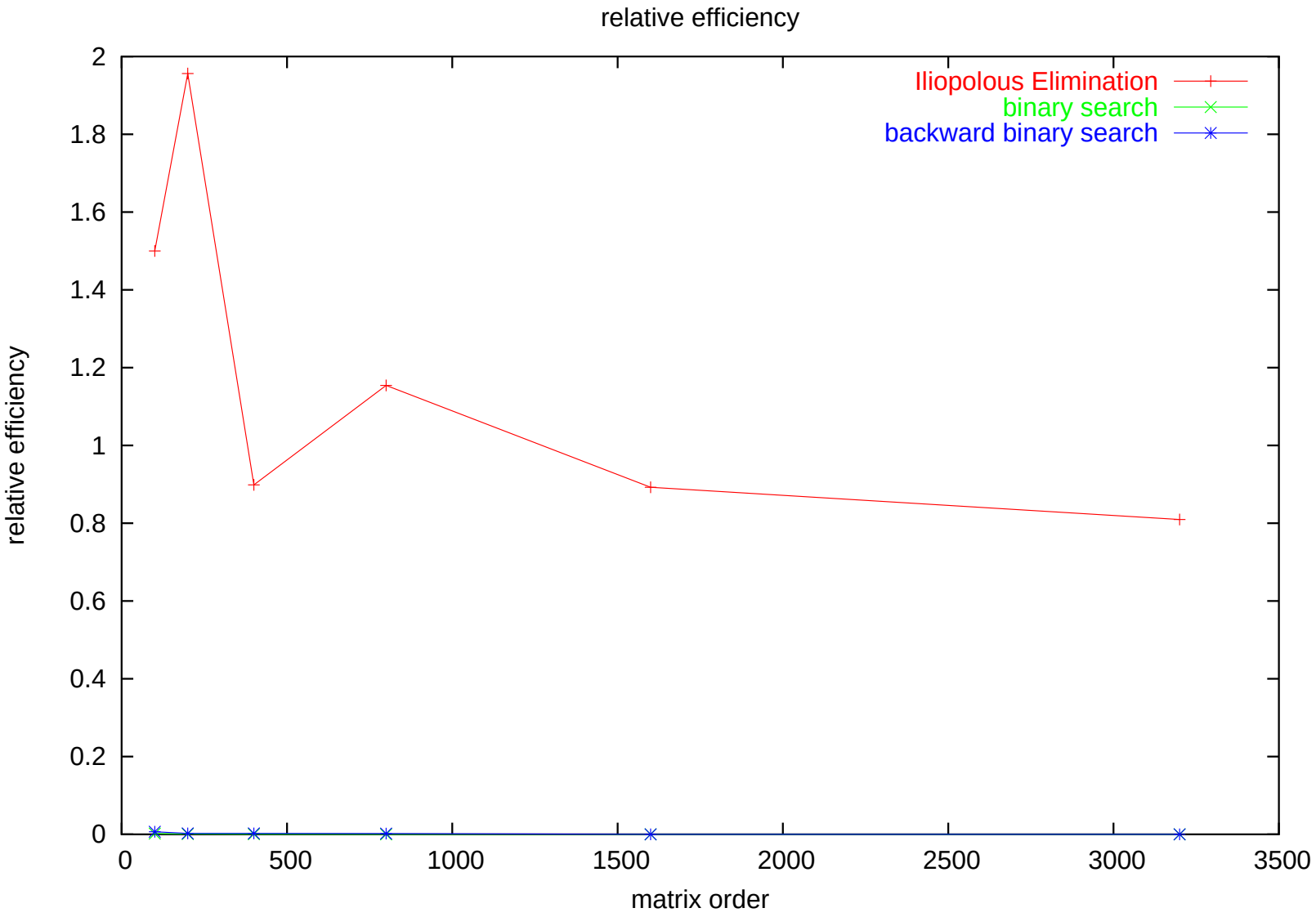

Rank
Degree
Minpoly
AA^T
<sqrt(n)
>= sqrt(n)
Valence
fail
Largest
Invariant
Factor, d
d_s, smooth part of d
d_r, rough part of d
Smooth Part
At each possible p
Starting with a possible e
Repeat unitl rank agrees, double e for next step
Rough Part
mod d_r
Bound Local
mod p^e
e = 1
e < sqrt(n)
else
Rank
Local
Smith
Form
EGV
Bisection
log d_r < n log n
log d_r >= n log n
Elimination
EGV
Bisection

Concocted example class fib

The matrix is randomly equivalent to $\oplus_{i=1}^{\tilde{}\log(n)} F_i$, where F_i is a $i \times i$ tridiagonal {0,1,-1} matrix whose det is fib(i).

SNF for n = 100 in 0.42sec: (1 96, 2 1, 30 1, 17160 1, 7368166325520 1,)

SNF for n = 200 in 2.23sec: (1 194, 2 1, 6 1, 30 1, 120 1, 181741560 1, 109877724068953573025813520 1,)

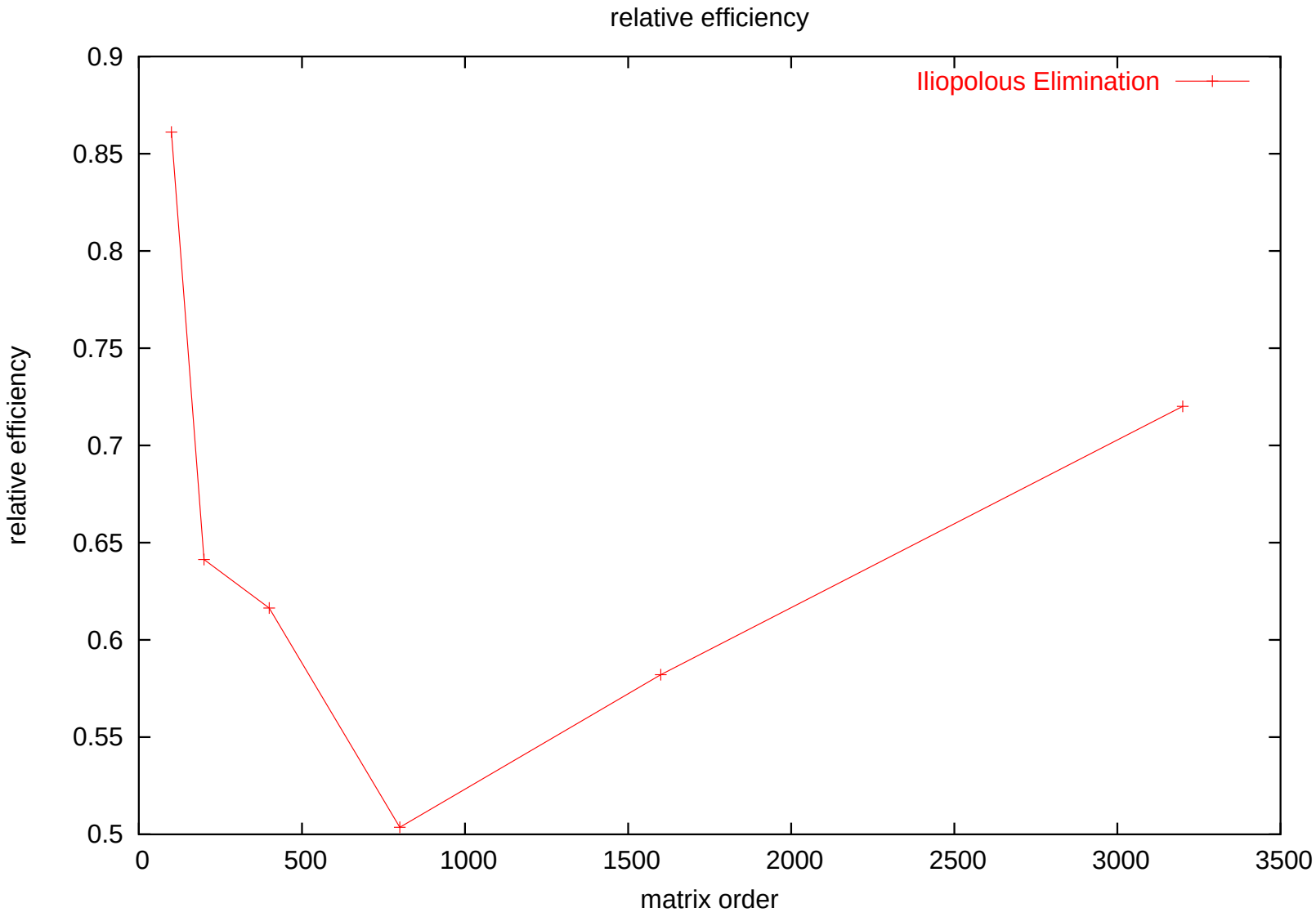

relative efficiency
Iliopolous Elimination
binary search
backward binary search
relative efficiency
matrix order
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
500
1000
1500
2000
2500
3000
3500

Concocted example class random

The matrix is randomly equivalent to $\mathrm{diag}(1, 2, 3, \ldots, n)$.

SNF for n = 100 in 0.93sec: (1 50, 2 17, 6 8, 12 5, 60 6, 420 2, 840 1, 2520 2, 27720 2, 360360 1, 720720 1, 232792560 1, 26771144400 1, 144403552893600 1, 309904450424599670640 0 1, 69720375229712477164533808935312303556800 1,)

SNF for n = 200 in 5.31sec: (1 100, 2 34, 6 16, 12 10, 60 12, 420 3, 840 3, 2520 4, 27720 3, 360360 3, 720720 1, 12252240 1, 232792560 2, 26771144400 1, 80313433200 1, 144403552893600 1, 5342931457063200 1, 309904450424599670640 0 1, 11822668841028222675113616000 1, 69720375229712477164533808935312303556800 1, 33729358883292626463946576679484140743239438278515723422884702191723401806067739 1,)


relative efficiency
Iliopolous Elimination
relative efficiency
matrix order
0.9
0.85
0.8
0.75
0.7
0.65
0.6
0.55
0.5
0
500
1000
1500
2000
2500
3000
3500

Concocted example class random-rough

The matrix is randomly equivalent to a diag built solely from rough primes.

SNF for n = 100 in 0.71sec: (1 82, 149 2, 20711 2, 2837407 2, 371700317 2, 7128096979109 2, 805474958639317 2, 87796770491685553 2, 9394254442610354171 2, 967608207588866479613 2,)

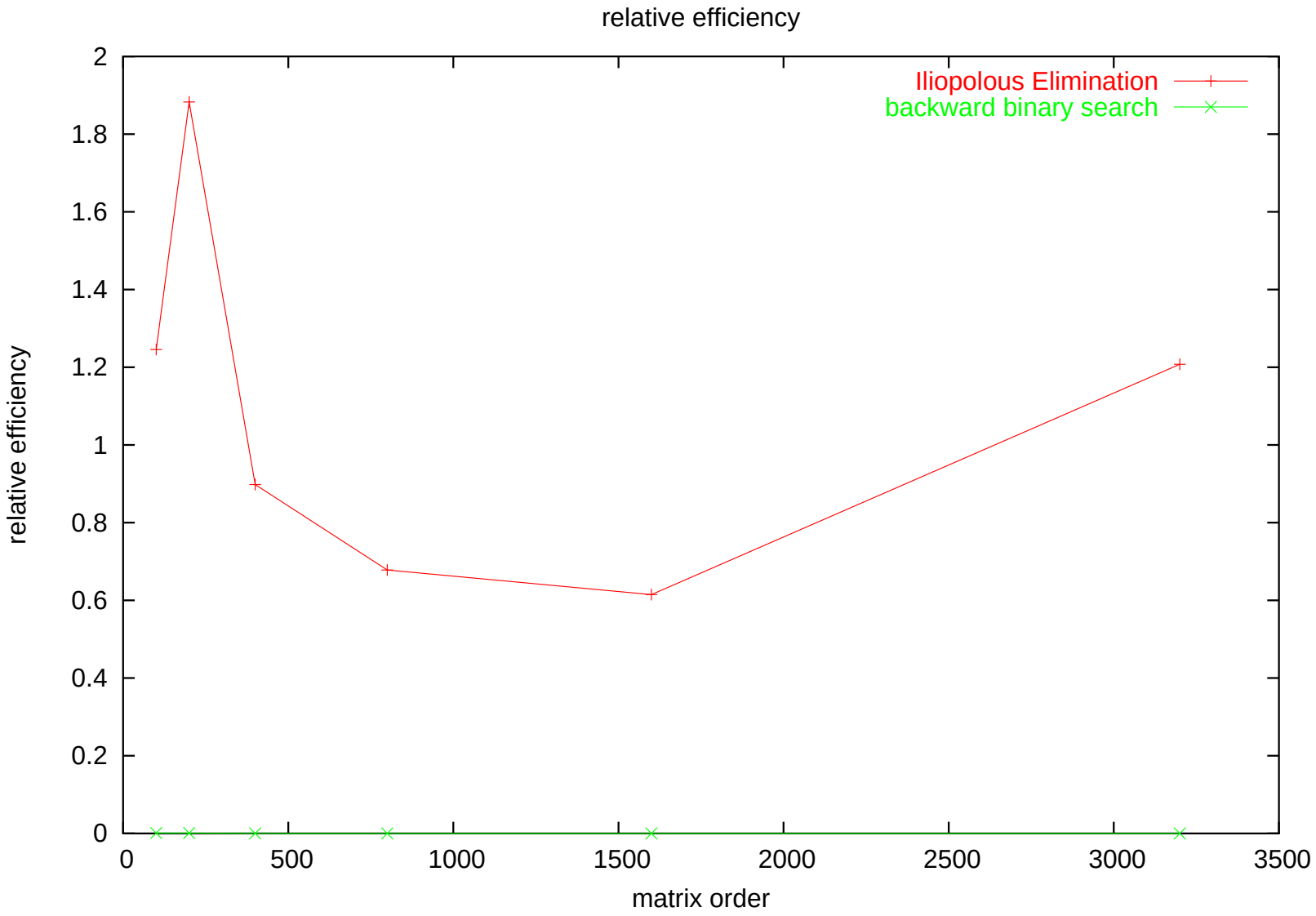
SNF for n = 200 in 8.02sec: (1 174, 167 2, 27221 2, 4273697 2, 645328247 2,

16634626222919 2, 2312213044985741 2, 316773187163046517 2,

41497287518359093727 2, 5270155514831604903329 2, 5955275731759713540761772,

64912505476180877594303293 2, 69456380859513539025904523512,

715400722852989451966816592153 2,)


relative efficiency
Iliopolous Elimination
backward binary search
relative efficiency
matrix order
0
0.2
0.4
0.6
0.8
1
1.2
1.4
1.6
1.8
2
0
500
1000
1500
2000
2500
3000
3500

Example matrices from other people's applications

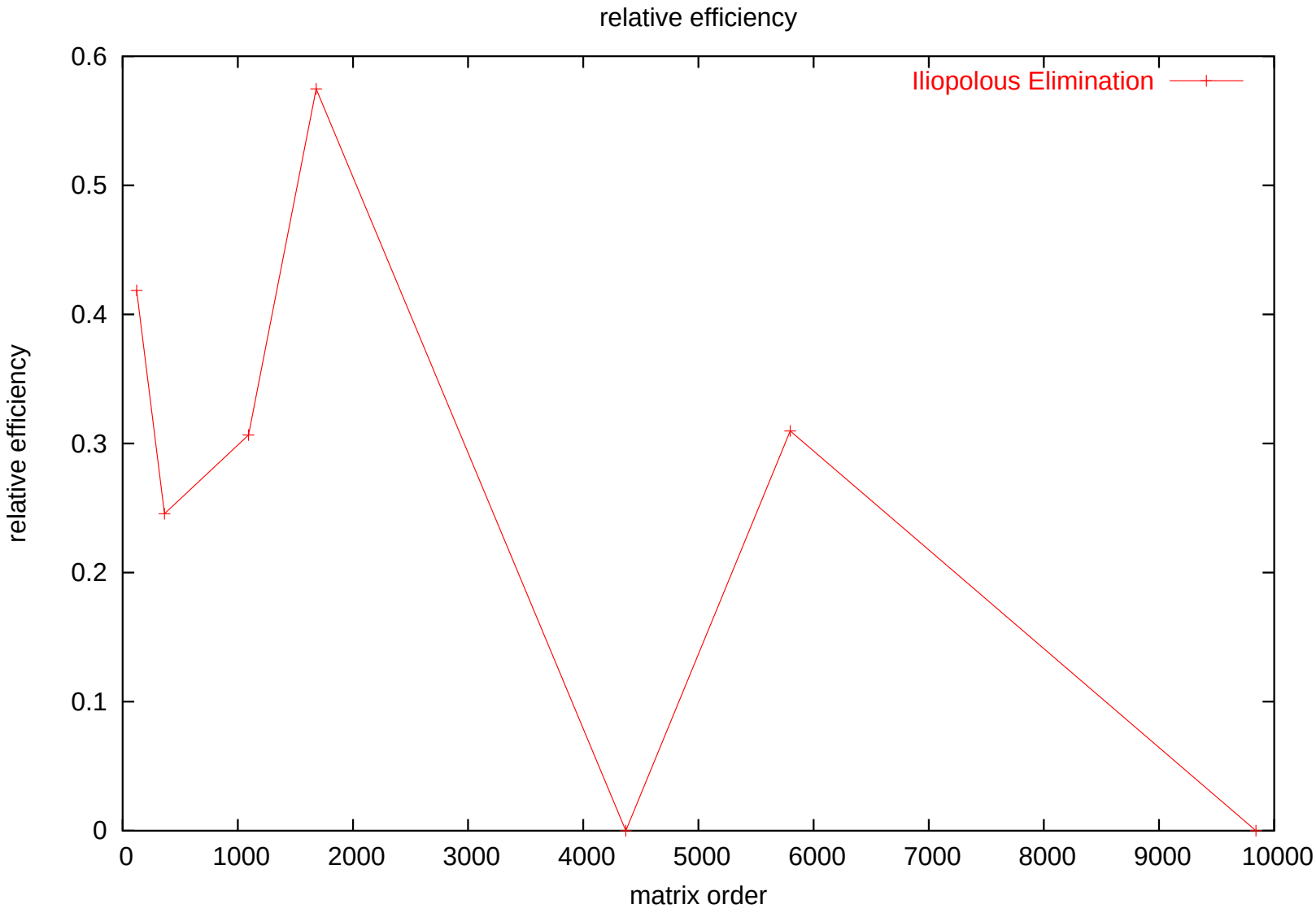
SNF of Chandler's 1093x1093 matrix in 47.03sec: (1 85, 3 266, 9 196, 27 196, 81 266, 243 83, 88452 1,)

SNF of Chandler's 9841x9841 matrix in 42639.8sec: (1 145, 3 1440, 9 1572, 27 1764, 81 1764, 243 1572, 729 1440, 2187 143, 7173360 1,)

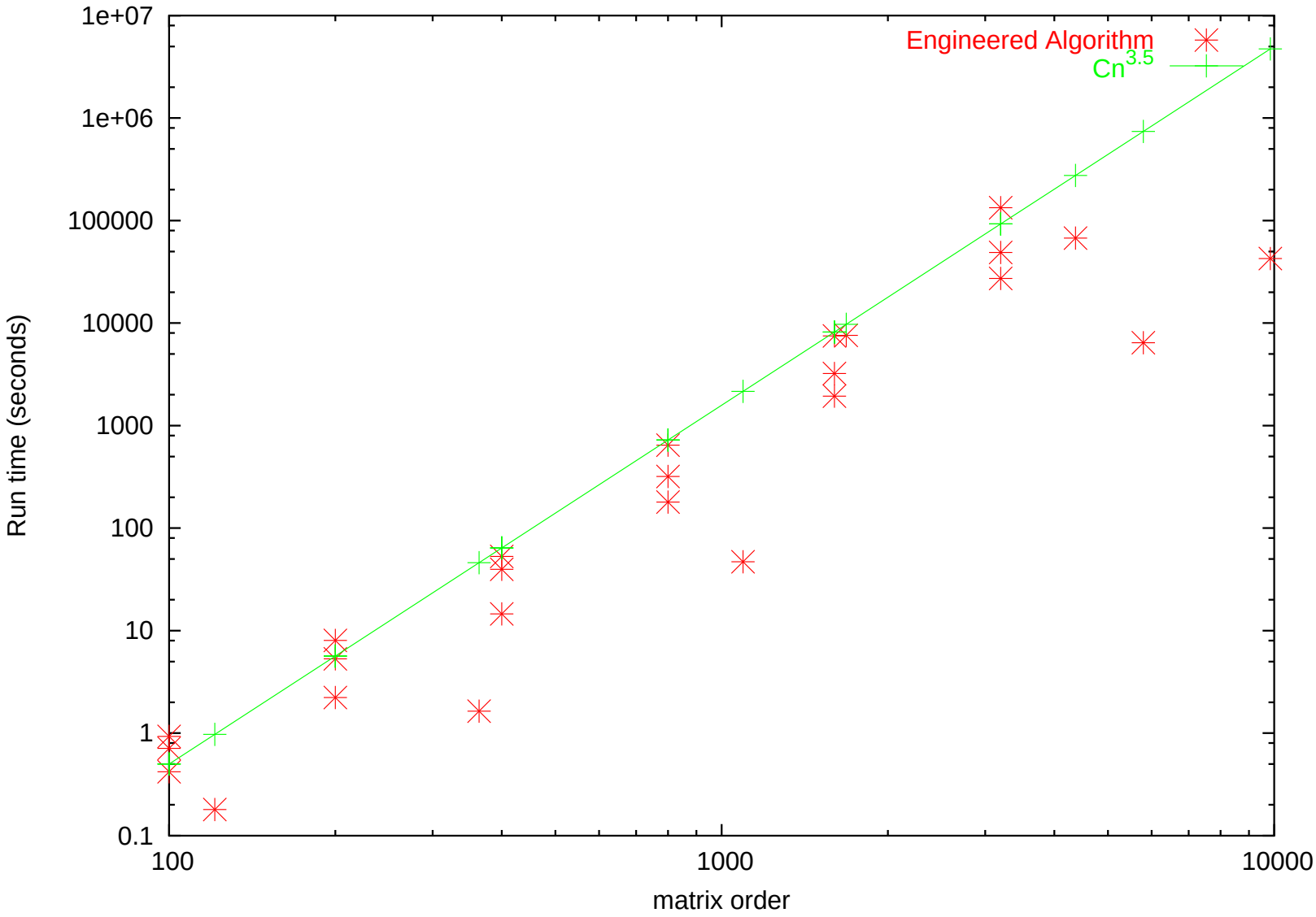
SNF of Chandler's 4369x70161 in 67558sec: (1 2801, 2 640, 4 416, 8 256, 16 255, 272 1,)

SNF of Krattenthaler's 5797x5797 matrix in 6445.45sec: (1 2226, 2 430, 4 801, 8 410, 16 1590, 80 170, 160 70, 320 99, 6720 1,)

Determinant via CRA of Krattenthaler's 5797x5797 matrix in 13050sec.


relative efficiency
Iliopolous Elimination
relative efficiency
matrix order
0
0.1
0.2
0.3
0.4
0.5
0.6
0
1000
2000
3000
4000
5000
6000
7000
8000
9000
10000

hybrid's runtimes



Last invariant factor

Outline:

1. Check if $\text{rank}(A) = n$. If no, return 0.
2. $s = 1$.
3. Repeat the following steps a few times
   - Choose random n-dimensional integer vector, $\vec{b}$
   - Solve $A\vec{x} = \vec{b}$ over the rational field, $\vec{x} = (x_1, ..., x_n)$ by Dixon lifing / our new solver.
   - $s' = \text{lcm}(\text{denominator}(x_i))$.
   - $s = \text{lcm}(s, s')$.
4. return s.

Bonus: In the non-singular case, with extra $O^{\sim}(n)$ bit operations, we can simultanously compute the $(n-1)$-th invariant factor.

Analysis

The algorithm returns the true last invariant factor of A with probability at least 1/3 for 2 iterations. The computing time is dominated by the solver. It requires $O^{\sim}(n^3)$ bit operations, if Dixon lifting used, $O^{\sim}(nE)$ for diagonally dominant $n \times n$ matrix with E entries.

The number of repetitions needed to get the k-rough part correct with probability of error less than ϵ is roughly
$reps = \log(1/\epsilon)/\log(k) + 1$.

k	reps	ϵ	k	reps	ϵ	k	reps	ϵ
1	2	0.333	1	11	10^{-3}	1	21	10^{-6}
10	2	0.1	10	4	10^{-3}	10	7	10^{-6}
40	2	0.025	40	3	10^{-3}	40	5	10^{-6}
n	2	$1/n$	1000	2	10^{-3}	1000000	2	10^{-6}

Computing arbitrary indexed i^{th} invariant factor

1. More efficient preconditioners

Outline:

1. $s = 0$.
2. Repeat the following steps a few times
   - Choose a random $i \times (n-i)$ matrix, L.
   - Choose a random $(n-i) \times i$ matrix, R.
   - Compute $A' = \begin{bmatrix} I_i & L \end{bmatrix} A \begin{bmatrix} I_i \\ R \end{bmatrix}$.
   - Compute $s' =$ last invariant factor of A'.
   - Compute $s = \gcd(s, s')$.
3. Return s.

The algorithm above will return the true i^{th} invariant factor of A with probability at least 1/2 for 10 iterations. It requires $O^{\sim}(n^3)$.

2. Tighter probability analysis

Theorem 1. *If both L and R are random matrices, with entries independently chosen from integer set of $[0, M-1]$, then for any prime p,*

$$\text{Prob}(p \nmid \frac{s_i(\begin{bmatrix} I_i & L \end{bmatrix} A \begin{bmatrix} I_i \\ R \end{bmatrix})}{s_i(A)}) \geq (\prod_{i=1}^{\infty}(1 - (\frac{1}{M}\lceil\frac{M}{p}\rceil)^i))^2$$

(about $(1 - 2/p)$.

Remark: Bad impact of small primes in last invariant factor's algorithm applies here also, even worse.

Summary

- Start with an engineered rank computation.
- Adaptively compute valence or rough part of largest invariant factor.
- If the latter, adaptively use a numeric-symbolic solver or tuned, engineered Dixon lifting.
- Bonus, also get rough part of next invariant factor from above step.
- In most cases, finish smoothly with local eliminations and ranks for a few tiny primes.
- In some cases, also eliminate with a rough modulus.
- But in rare cases of enormous rough part, conduct the EGV binary search roughly.

- Open: Good fast, memory efficient multi-algorithm for sparse matrix Smith form.
- Open: Semi-automatic retuning of engineered algorithms.

References

[1] W. Eberly, M. Giesbrecht, and G. Villard. On computing the determinant and smith form of an integer matrix. In *Proc. 41st FOCS*, pages 675 – 687, 2000.

[2] M. Giesbrecht. Probabilistic computation of the smith normal form of a dense integer matrix. In *Proc. ISSAC'95*, pages 1 – 1. ACM Press, 1996.

[3] J. L. Hafner and K. S. McCurley. Asymptotically fast triangularization of matrices over rings. *Siam J. Comput.*, 20:1068–1083, 1991.

[4] C. Iliopoulos. Worst-case complexity bounds on algorithms for computing the canonical structure of finite abelian groups of the hermite and smith normal forms of an integer matrix. *SIAM J. Comput.*, Vol. 18, No.4:658 – 669, 1989.

[5] E. Kaltofen and G. Villard. On the complexity of computing determinants. Technical Report 36, Laboratoire de l'Informatique du Parallélisme, Ecole Normale Supérieure de Lyon, October 2003.

[6] R. Kannan and A. Bachem. Polynomail algorithm for computing the smith and hermite normal forms od an integer matrix. *SIAM J. Comput.*, Vol8, No.4:499 – 507, 1979.

[7] H. M. S. Smith. On systems of indeterminate equations and congruences. *Philos. Trans.*, pages 293–326, 1861.

[8] A. Storjohann. Near optimal algorithms for computing smith normal forms of integer matrices. In *Proc. ISSAC'96*, pages 267 – 274. ACM Press, 1996.