SuperSPAR

An Integer Factoring Algorithm

Maxwell Sayles

Department of Computer Science

University of Calgary

May 24, 2013

Overview

What is SuperSPAR?

- Improves upon SPAR (named after Shanks, Pollard, Atkin, and Rickert).

Overview

What is SuperSPAR?

- Improves upon SPAR (named after Shanks, Pollard, Atkin, and Rickert).
- Based on arithmetic in the ideal class group of imaginary quadratic number fields.

Overview

What is SuperSPAR?

- Improves upon SPAR (named after Shanks, Pollard, Atkin, and Rickert).
- Based on arithmetic in the ideal class group of imaginary quadratic number fields.
- Uses the order of an element in this group to factor an integer.

Binary Quadratic Forms

Ideal class group originally studied by Gauss.

Binary Quadratic Forms

Ideal class group originally studied by Gauss.

- $Q(x, y) = ax^2 + bxy + cy^2$

Binary Quadratic Forms

Ideal class group originally studied by Gauss.

- $Q(x, y) = ax^2 + bxy + cy^2$
- $Q(x, y) = 2x^2 + xy + 18y^2$
  $= \{2, 8, 18, 19, 21, 24, 28, 32, 33, 39, 46, 50, 54, 63, 72, ...\}$

Binary Quadratic Forms

Ideal class group originally studied by Gauss.

- $Q(x, y) = ax^2 + bxy + cy^2$
- $Q(x, y) = 2x^2 + xy + 18y^2$
  $= \{2, 8, 18, 19, 21, 24, 28, 32, 33, 39, 46, 50, 54, 63, 72, ...\}$
- $\Delta = b^2 - 4ac$

Binary Quadratic Forms

Ideal class group originally studied by Gauss.

- $Q(x, y) = ax^2 + bxy + cy^2$
- $Q(x, y) = 2x^2 + xy + 18y^2$
  $= \{2, 8, 18, 19, 21, 24, 28, 32, 33, 39, 46, 50, 54, 63, 72, ...\}$
- $\Delta = b^2 - 4ac$
- $\Delta = 1^2 - 4 \cdot 2 \cdot 18 = -143$

Binary Quadratic Forms

Ideal class group originally studied by Gauss.

- $Q(x, y) = ax^2 + bxy + cy^2$
- $Q(x, y) = 2x^2 + xy + 18y^2$
  $= \{2, 8, 18, 19, 21, 24, 28, 32, 33, 39, 46, 50, 54, 63, 72, ...\}$
- $\Delta = b^2 - 4ac$
- $\Delta = 1^2 - 4 \cdot 2 \cdot 18 = -143$
- Extremely rich theory.

Binary Quadratic Forms

Ideal class group originally studied by Gauss.

- $Q(x, y) = ax^2 + bxy + cy^2$
- $Q(x, y) = 2x^2 + xy + 18y^2$
  $= \{2, 8, 18, 19, 21, 24, 28, 32, 33, 39, 46, 50, 54, 63, 72, ...\}$
- $\Delta = b^2 - 4ac$
- $\Delta = 1^2 - 4 \cdot 2 \cdot 18 = -143$
- Extremely rich theory.
- Different properties when $\Delta < 0$ or $\Delta > 0$.

Equivalence of Forms

Two forms are equivalent

$$Q(x, y) \sim Q'(x', y')$$

if there exists an invertible integral linear transformation

$$x = \alpha x' + \beta y'$$
$$y = \gamma x' + \delta y'.$$

Equivalence of Forms

Substituting and simplifying give

$$\begin{aligned} ax^2 + bxy + cy^2 &= a(\alpha x' + \beta y')^2 \\ &\quad + b(\alpha x' + \beta y')(\gamma x' + \delta y') \\ &\quad + c(\gamma x' + \delta y')^2 \\ &= a'x'^2 + b'x'y' + c'y'^2 \end{aligned}$$

Example of Equivalent Forms

$$92x^2 - 26xy + 6y^2 \sim 6x'^2 + 2x'y' + 64y'^2$$

is given by the transformation

$$\begin{bmatrix} \alpha & \beta \\ \gamma & \delta \end{bmatrix} = \begin{bmatrix} 2 & 1 \\ 1 & 0 \end{bmatrix}$$

Equivalence Class of Binary Quadratic Forms

An *equivalence class* is the set of all forms equivalent to a given form.

Similar to modular arithmetic:

$$0 \equiv 7, 14, 21, 28, 35, 42, \ldots \pmod 7$$
$$1 \equiv 8, 15, 22, 29, 36, 43, \ldots \pmod 7$$
$$2 \equiv 9, 16, 23, 30, 37, 44, \ldots \pmod 7$$
$$3 \equiv 10, 17, 24, 31, 38, 45, \ldots \pmod 7$$
$$4 \equiv 11, 18, 25, 32, 39, 46, \ldots \pmod 7$$
$$5 \equiv 12, 19, 26, 33, 40, 47, \ldots \pmod 7$$
$$6 \equiv 13, 20, 27, 34, 41, 48, \ldots \pmod 7$$

Reduced Forms

Work in reduced forms:

$$-a < b \leq a < c \text{ or } 0 \leq b \leq a = c.$$

Bounds:

$$-\Delta = 4ac - b^2 \geq 4ac - a^2 \geq 3a^2$$

gives

$$|b| \leq a \leq \sqrt{|\Delta|/3} \text{ and } c \leq |\Delta|/4.$$

* Only holds for $\Delta < 0$.

Ideal Reduction

Input: (a, b, c) and Δ.

1: **while** $a > c$ or $b > a$ or $b \leq -a$ **do**
2: **if** $a > c$ **then**
3: swap a with c and set $b \leftarrow -b$
4: **if** $b > a$ or $b \leq -a$ **then**
5: $b \leftarrow b'$ such that $-a < b' \leq a$ and $b' \equiv b \pmod{2a}$
6: $c \leftarrow (b^2 - \Delta)/4a$
7: **if** $a = c$ and $b < 0$ **then**
8: $b \leftarrow -b$
9: **return** (a, b, c)

REDUCTION EXAMPLE

$$
\begin{aligned}
Q(x, y) &= 72x^2 + 130xy + 64y^2 \\
&\sim 64x^2 - 130xy + 72y^2 \\
&\sim 64x^2 - 2xy + 6y^2 \\
&\sim 6x^2 + 2xy + 64y^2
\end{aligned}
$$

Multiplication of Forms

$$
\begin{aligned}
Q_1(x,y) &= 2x^2 + xy + 18y^2 \\
&= \{2, 8, 18, 19, 21, 24, 28, 32, 33, 39, 46, 50, 54, 63, 72, ...\} \\
Q_2(x,y) &= 3x^2 + xy + 12y^2 \\
&= \{3, 12, 14, 16, 22, 26, 27, 36, 42, 48, 49, 53, 56, 64, 69, ...\}
\end{aligned}
$$

$$
\begin{aligned}
R(x,y) &= Q_1(x,y) \cdot Q_2(x,y) \\
&= 6x^2 + xy + 6y^2 \\
&= \{6, 24, 28, 32, 44, 52, 54, 57, 63, 72, 84, 96, 98, 99, 112...\}
\end{aligned}
$$

Ideal Multiplication

Input: Ideals (a_1, b_1, c_1), (a_2, b_2, c_2), and Δ.

1: $s = \gcd(a_1, a_2, (b_1 + b_2)/2) = Ya_1 + Va_2 + W(b_1 + b_2)/2$
2: $U = (V(b_1 - b_2)/2 - Wc_2) \bmod (a_1/s)$
3: $a = (a_1 a_2)/s^2$
4: $b = (b_2 + 2Ua_2/s) \bmod 2a$
5: $c = (b^2 - \Delta)/4a$

Split XGCD

Split $s = \gcd(a_1, a_2, (b_1 + b_2)/2) = Ya_1 + Va_2 + W(b_1 + b_2)/2$ into two computations.

Input: Ideals (a_1, b_1, c_1), (a_2, b_2, c_2), and Δ.

1: $s' = \gcd(a_1, a_2) = Y'a_1 + V'a_2$ $\{Y'$ is never used$\}$
2: **if** $s' = 1$ **then**
3: $\quad s = 1, V = V', W = 0$
4: **else**
5: $\quad s = \gcd(s', (b_1 + b_2)/2) = V''s' + W(b_1 + b_2)/2$
6: $\quad V = V'V''$
7: ...

Simultaneous Multiplication and Reduction

NUCOMP – due to Shanks with improvements by Jacobson, Williams, Imbert, and Schmidt.

- $a_1, a_2 \leq \sqrt{|\Delta|/3}$, but the product $a \leq |\Delta|/3$.

Simultaneous Multiplication and Reduction

NUCOMP – due to Shanks with improvements by Jacobson, Williams, Imbert, and Schmidt.

- $a_1, a_2 \leq \sqrt{|\Delta|/3}$, but the product $a \leq |\Delta|/3$.
- Reduction is the simple continued fraction expansion of $b/2a$.

Simultaneous Multiplication and Reduction

NUCOMP – due to Shanks with improvements by Jacobson, Williams, Imbert, and Schmidt.

- $a_1, a_2 \leq \sqrt{|\Delta|/3}$, but the product $a \leq |\Delta|/3$.
- Reduction is the simple continued fraction expansion of $b/2a$.
- $b/2a \approx sU/a_1$.

Simultaneous Multiplication and Reduction

NUCOMP – due to Shanks with improvements by Jacobson, Williams, Imbert, and Schmidt.

- $a_1, a_2 \leq \sqrt{|\Delta|/3}$, but the product $a \leq |\Delta|/3$.
- Reduction is the simple continued fraction expansion of $b/2a$.
- $b/2a \approx sU/a_1$.
- Derive the reduced product from the simple continued fraction expansion.

Simultaneous Multiplication and Reduction

NUCOMP – due to Shanks with improvements by Jacobson, Williams, Imbert, and Schmidt.

- $a_1, a_2 \leq \sqrt{|\Delta|/3}$, but the product $a \leq |\Delta|/3$.
- Reduction is the simple continued fraction expansion of $b/2a$.
- $b/2a \approx sU/a_1$.
- Derive the reduced product from the simple continued fraction expansion.
- Simplifies when squaring and cubing.

Identity

Denoted 1_G:

$$(a, b, c) = \begin{cases} (1, 0, \Delta/4) & \text{when } \Delta \equiv 0 \pmod 4 \\ (1, 1, (1-\Delta)/4) & \text{when } \Delta \equiv 1 \pmod 4 \end{cases}$$

Inverse

$$(a, b, c)^{-1} = (a, -b, c)$$

Class Group of Binary Quadratic Forms

Class group is

- closed,
- associative,
- commutative,
- has identity,
- has inverse

– finite Abelian group.

How Does this Factor an Integer?

Ambiguous form is any form q such that $q^2 = 1_G$.

Three varieties:

$$(a, 0, c) \Rightarrow \Delta = -4ac$$
$$(a, a, c) \Rightarrow \Delta = a^2 - 4ac = a(a - 4c)$$
$$(a, b, a) \Rightarrow \Delta = b^2 - 4a^2 = (b + 2a)(b - 2a)$$

Choose some form with $\Delta = -kN$.

Finding an Ambiguous Form

Let $h = \text{ord}(q)$ such that $q^h = 1_G$.

If h is even, then let h' be the odd part of h and

$$q^{h'\left(2^j\right)}$$

is an ambiguous form.

How Do We Find the Order?

- Exponentiate q^E for $E = \prod {p_i}^{e_i}$ for odd primes p_i.
- Use an order finding algorithm.

Improvements to Group Arithmetic

- Implementations for 64-bit and 128-bit arithmetic.
- Full/Partial extended greatest common divisor.
  - Implementations of XGCD for 32-bit, 64-bit, 128-bit arithmetic.

How to Compute XGCD?

Start with

$$s = Ua + Vb$$
$$t = Xa + Yb$$

with

$$\begin{bmatrix} s & U & V \\ t & X & Y \end{bmatrix} \leftarrow \begin{bmatrix} a & 1 & 0 \\ b & 0 & 1 \end{bmatrix}.$$

Maintain $s \geq t \geq 0$.

Extended Euclidean Algorithm

1: **procedure** $\mathrm{EEA}(a, b)$ $\qquad \{a, b \in \mathbb{Z}_{\geq 0}\}$

2: $\quad \begin{bmatrix} s & U & V \\ t & X & Y \end{bmatrix} \leftarrow \begin{bmatrix} a & 1 & 0 \\ b & 0 & 1 \end{bmatrix}$

3: $\quad$ **while** $t > 0$ **do**

4: $\quad\quad q \leftarrow \lfloor s/t \rfloor$

5: $\quad\quad$ subtract q times the second row from the first

6: $\quad\quad$ swap rows so that $s \geq t$

7: $\quad$ **return** (s, U, V)

Our Left-to-Right Binary XGCD

1: **procedure** OUR_L2R_XGCD(a, b) $\{a, b \in \mathbb{Z}_{\geq 0}\}$

2: $\begin{bmatrix} s & U & V \\ t & X & Y \end{bmatrix} \leftarrow \begin{bmatrix} a & 1 & 0 \\ b & 0 & 1 \end{bmatrix}$

3: **while** $t > 0$ **do**

4: $k \leftarrow$ NUMBITS(s) - NUMBITS(t)

5: subtract 2^k times the second row from the first

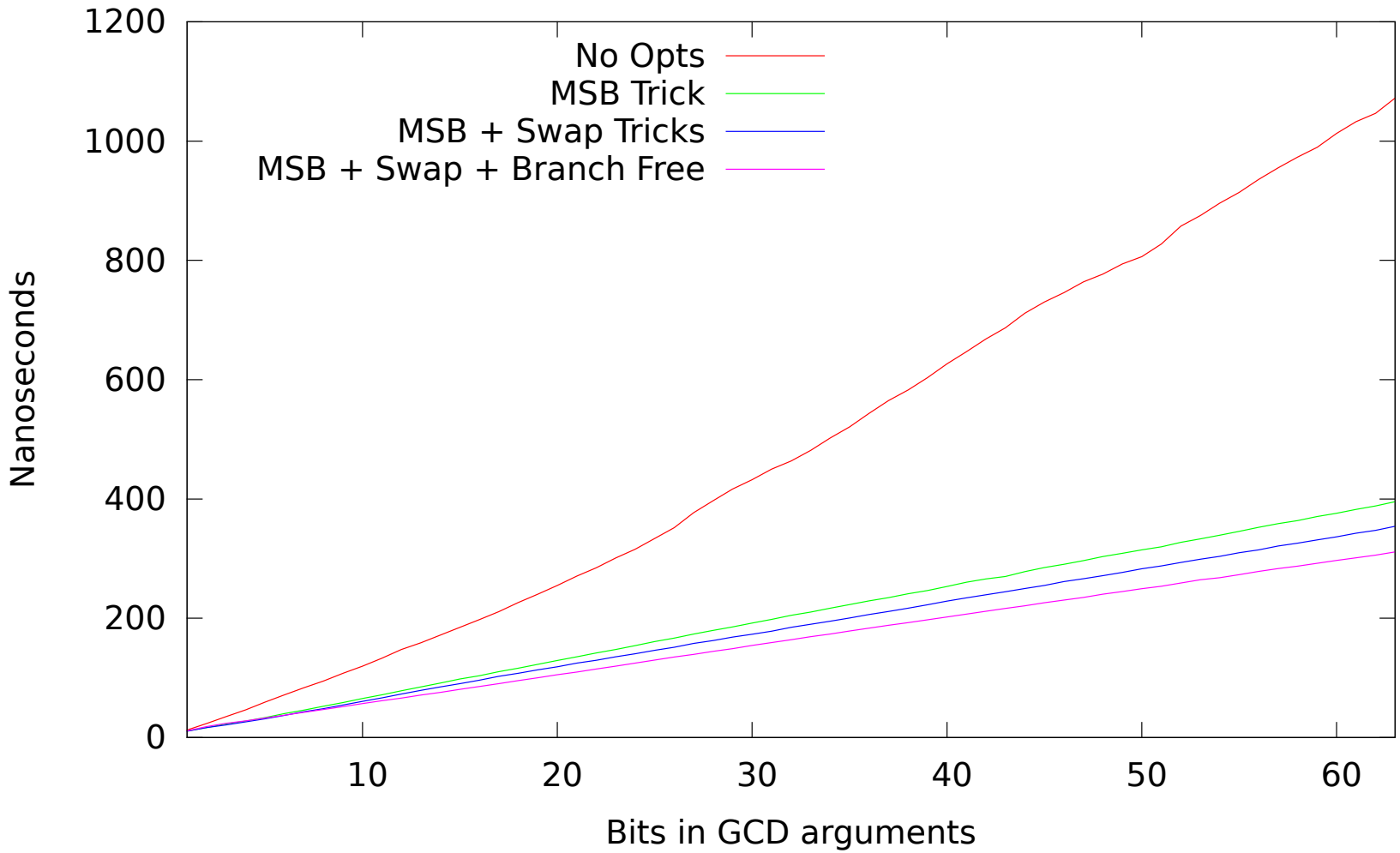
6: **if** $s < 0$ **then** negate the first row

7: **if** $s < t$ **then** swap the rows of the matrix

8: **return** (s, U, V)

Binary XGCD Optimizations

64-bit simplified left-to-right binary XGCD



XGCD Improvement

How much faster than the reference implementations?

Bit Range	Algorithm	GMP	Pari	Flint
1 – 31	EEA	3.44	1.60	1.17
32 – 63	Our L2R Binary	1.94	1.29	1.16
64 – 118	Our L2R Binary	1.38	1.38	–
119 – 127	Lehmer w/ Our 64-bit L2R	1.12	1.13	–

All times are based on the average. The measurement of improvement is an average over the bit range.

Class Group Improvements

How much faster than the reference implementations?

Operation	Bit Range	Pari	GMP
Multiplication	1 – 59	4.27	5.66
Squaring	1 – 59	4.27	4.95
Cubing	1 – 59	5.86	4.10
Multiplication	59 – 118	3.60	2.74
Squaring	59 – 118	4.02	2.66
Cubing	59 – 118	3.82	1.83

All times are based on the average. The measurement of improvement is an average over the bit range.

Exponentiation Stage of SuperSPAR

Exponentiate a random form to the product of many odd prime powers.

$$E = \prod {p_i}^{e_i}$$

2,3 Representations

$$N = \sum_{i=0}^{k} s_i 2^{a_i} 3^{b_i}.$$

Example:

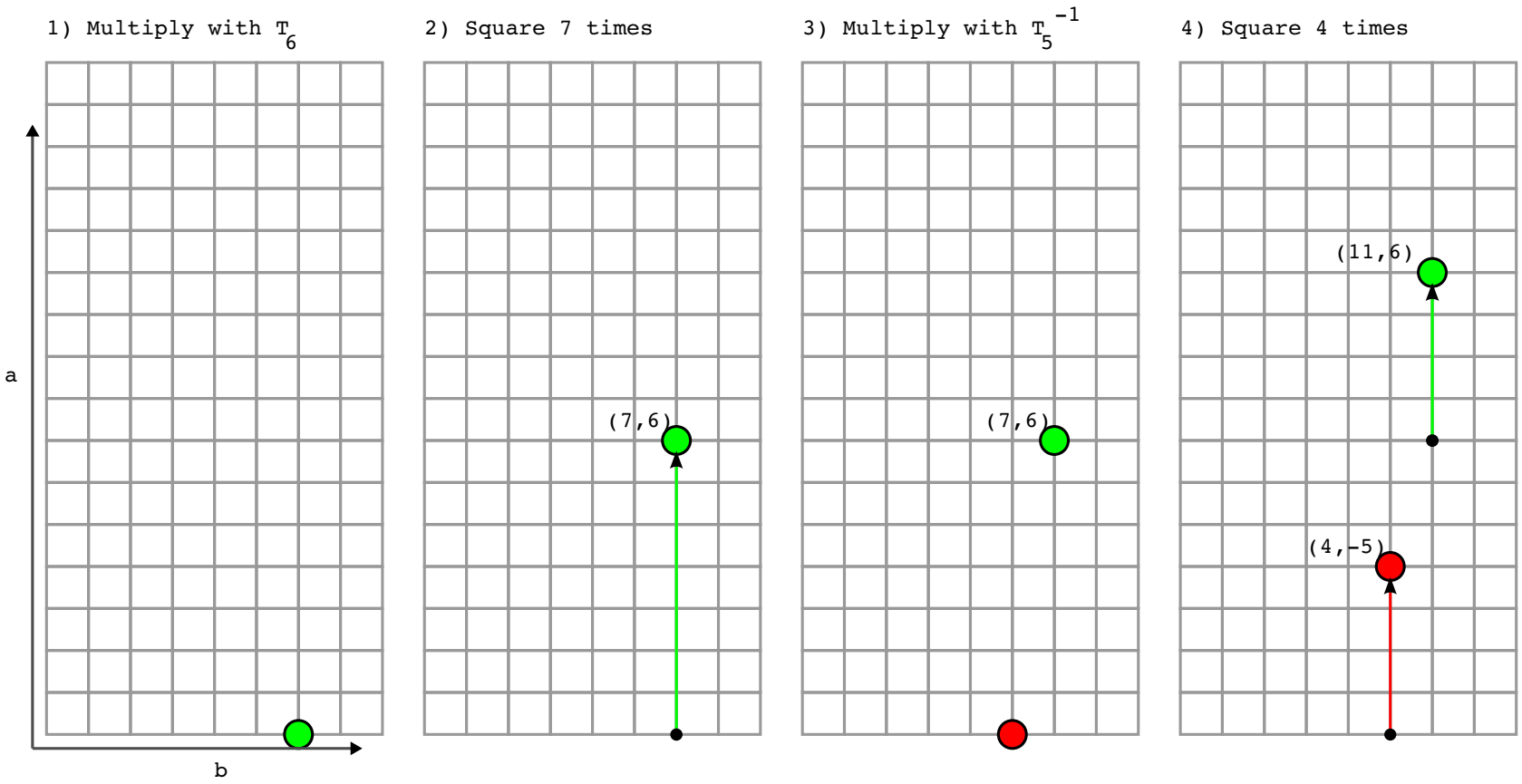
- $23814216 = 2^3 + 2^6 + 2^{13} + 2^{14} + 2^{16} + 2^{17} + 2^{19} + 2^{21} + 2^{22} + 2^{24}$
- $23814216 = 2^3 3^3 - 2^4 3^5 + 2^5 3^6 + 2^7 3^7 + 2^9 3^8 + 2^{10} 3^9$
- $23814216 = 2^3 3^3 - 2^4 3^6 - 2^8 3^5 + 2^{15} 3^6$
- $23814216 = 2^3 3^2 - 2^{13} 3^2 + 2^{15} 3^6$

How to Exponentiate 2,3 Representations?

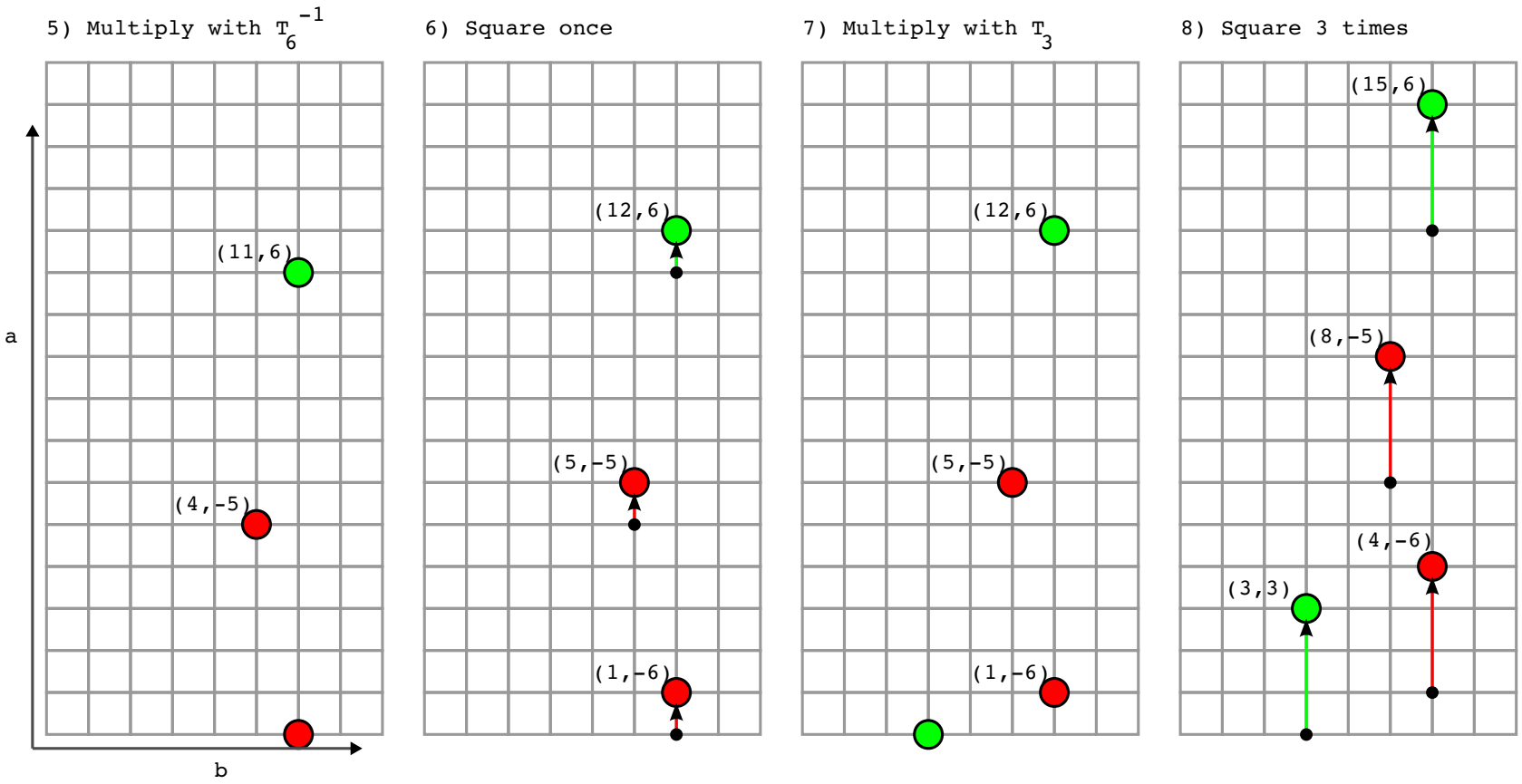
First we nest common factors of 2:

$$2^{15}3^6 - 2^8 3^5 - 2^4 3^6 + 2^3 3^3 = (((3^6 2^7 - 3^5)2^4 - 3^6)2^1 + 3^3)2^3$$

Exponentiating $g^{2^{15}3^6-2^8 3^5-2^4 3^6+2^3 3^3}$



Exponentiating $g^{2^{15}3^6-2^8 3^5-2^4 3^6+2^3 3^3}$



Left-to-Right Best Approximations

Left-to-Right best approximations is a technique combining the greedy approach of Berthé and Imbert with the tree based approach of Doche and Habsieger.

Left-to-Right Best Approximations

Left-to-Right best approximations is a technique combining the greedy approach of Berthé and Imbert with the tree based approach of Doche and Habsieger.

- Iterate on a set of partial representations $x + \sum s_i 2^{a_i} 3^{b_i}$.

Left-to-Right Best Approximations

Left-to-Right best approximations is a technique combining the greedy approach of Berthé and Imbert with the tree based approach of Doche and Habsieger.

- Iterate on a set of partial representations $x + \sum s_i 2^{a_i} 3^{b_i}$.
- Keep the best $2^a 3^b$ approximations for each x.

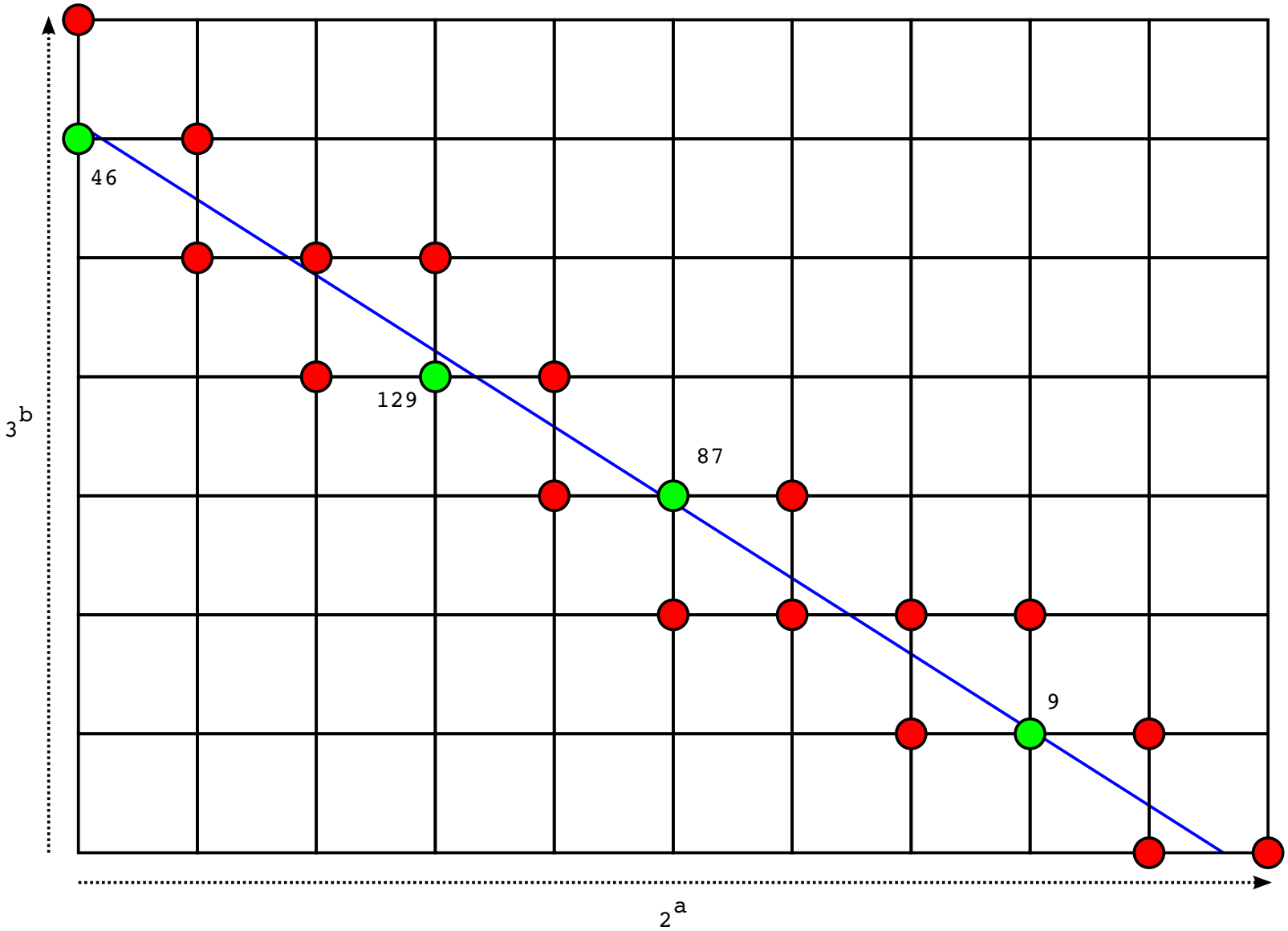
Left-to-Right Best Approximations

Left-to-Right best approximations is a technique combining the greedy approach of Berthé and Imbert with the tree based approach of Doche and Habsieger.

- Iterate on a set of partial representations $x + \sum s_i 2^{a_i} 3^{b_i}$.
- Keep the best $2^a 3^b$ approximations for each x.
- Squares and cubes are bound $a \leq A$, $b \leq B$, and the bounds A and B are iterated.

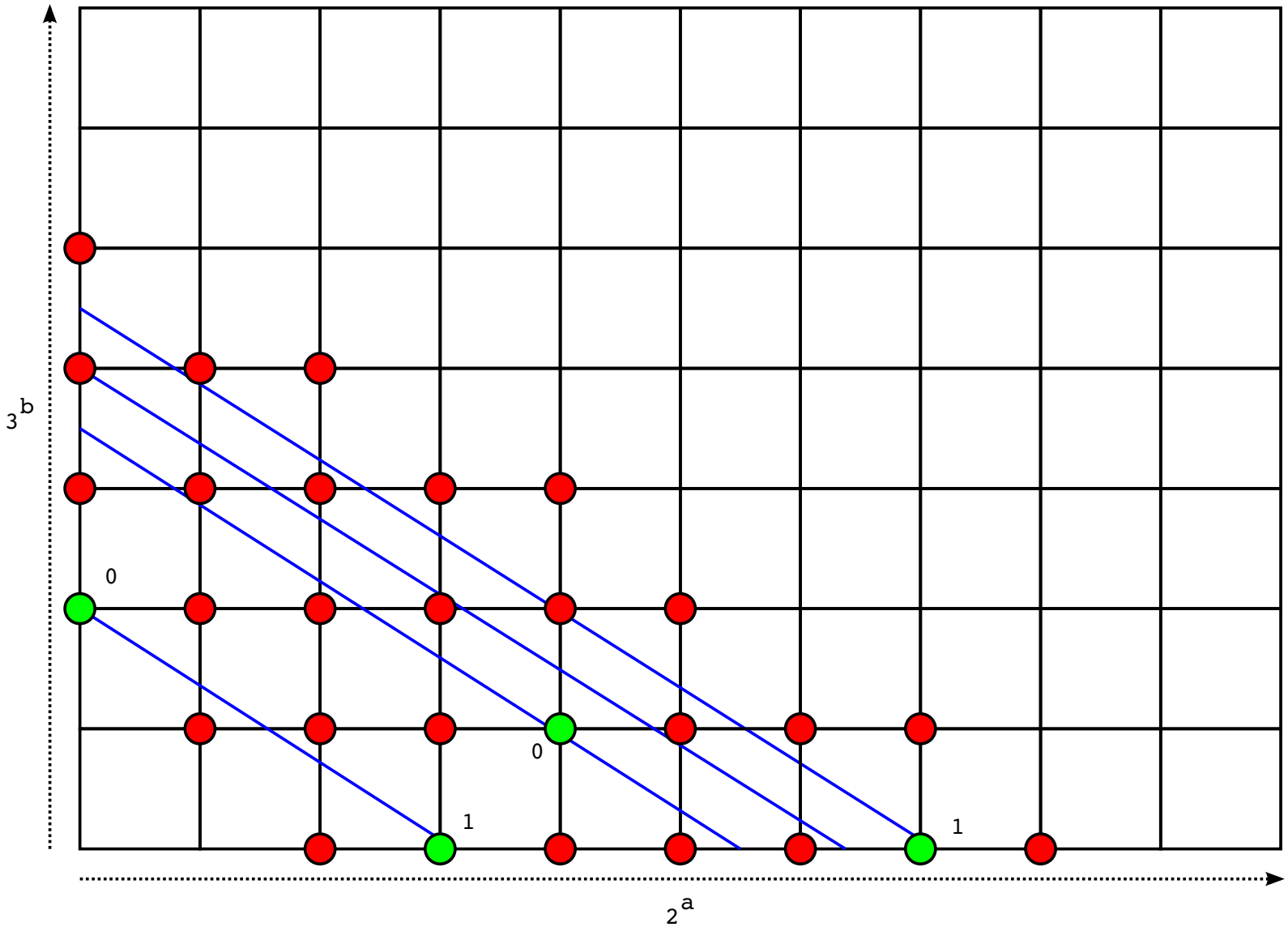
Left-to-Right Best Approximations

Example – First Iteration on 777



Left-to-Right Best Approximations

Example – Second Iteration on 9, 48, 87, and 129



2,3 Representations for $E = 777$:

$2^8 3^1 + 2^0 3^2$	8 squares, 2 cubes, 1 multiplication
$2^0 3^6 + 2^4 3^1$	4 squares, 6 cubes, 1 multiplication
$2^8 3^1 + 2^3 3^0 + 2^0 3^0$	8 squares, 1 cube, 2 multiplications
$2^3 3^4 + 2^7 3^0 + 2^0 3^0$	7 squares, 4 cubes, 2 multiplications

$2^8 3^1 + 2^3 3^0 + 2^0 3^0$ possibly preferred since multiplication is cheaper than squaring.

Exponentiation Improvement

How much faster than the reference implementations?

	Binary	NAF	Pruned Tree	Greedy Left-to-Right
Left-to-Right Best Approximations (64-bit)	1.40	1.25	1.13	1.009
Left-to-Right Best Approximations (128-bit)	1.42	1.25	1.18	1.010

All times are based on the average. The measurement of improvement is an average over the bit range.

After Exponentiating

Exponentiation stage computes:

$$q' = q^E.$$

Then repeatedly square q'.

Search Stage of SuperSPAR

Use a bounded primorial steps algorithm to find the order of an element (due to Sutherland).

Primorials

Primorial:

$$P_t = \prod_{p_i \leq p_t} p_i.$$

Coprime to P_t:

$$\phi(P_t) = \prod_{p_i \leq p_t} (p_i - 1).$$

Baby-Step Giant-Step

Order finding due to Shanks. Requires $O(\sqrt{N})$.

Input: A group element g, and a bound on the order N.

1: $s \leftarrow \sqrt{N}$
2: **for** $i = 1, 2, 3, ..., s$ **do**
3: store $g^i \mapsto i$ in a table
4: **if** $g^i = 1_G$ **then return** i
5: **for** $j = 1, 2, 3, ..., s$ **do**
6: **if** g^{sj} is in the lookup table **then**
7: then there exists some $g^i = g^{sj}$
8: **return** $sj - i$

What if the Order is Odd?

Sutherland noticed for odd orders:

Input: A group element g, and a bound on the order N.

1: $s \leftarrow \sqrt{N/2}$ with s even
2: **for** $i = 1, 3, 5, ..., s$ **do**
3: store $g^i \mapsto i$ in a table
4: **if** $g^i = 1_G$ **then return** i
5: **for** $j = 1, 2, 3, ..., s$ **do**
6: **if** g^{sj} is in the lookup table **then**
7: then there exists some $g^i = g^{sj}$
8: **return** $sj - i$

If Both 2 and 3 Do Not Divide the Order?

Why stop at odd orders?

Input: A group element g, and a bound on the order N.

1: $s \leftarrow \sqrt{N/3}$ with s a multiple of 6
2: **for** $i = 1, 5, 7, 11, ..., s$ **do**
3: store $g^i \mapsto i$ in a table
4: **if** $g^i = 1_G$ **then return** i
5: **for** $j = 1, 2, 3, ..., s$ **do**
6: **if** g^{sj} is in the lookup table **then**
7: then there exists some $g^i = g^{sj}$
8: **return** $sj - i$

General Case

For all primes up to p_t.

Input: A group element g, and a bound on the order N.

1: $s \leftarrow \sqrt{N/(P_t/\phi(P_t))}$ with s a multiple of P_t
2: **for** $i = 1, ..., s$ with i coprime to P_t **do**
3: store $g^i \mapsto i$ in a table
4: **if** $g^i = 1_G$ **then return** i
5: **for** $j = 1, 2, 3, ..., s$ **do**
6: **if** g^{sj} is in the lookup table **then**
7: then there exists some $g^i = g^{sj}$
8: **return** $sj - i$

How is This Useful?

The exponentiation stage computed

$$q'' = q^{E\left(2^j\right)}$$

for

$$E = \prod_{i=2}^{t} {p_i}^{e_i}.$$

The order of q'' is likely coprime to E.

How to Pick the Exponent and the Order Bound

Exponent: Let $p_t \approx N^{1/2r}$ and

$$E = \prod_{i=2}^{t} {p_i}^{e_i}$$

where $e_i = \log_{p_i} {p_t}^2$.

Search bound: Maximize w such that $m\phi(P_w) \approx p_t$.

How to pick r ?

Smoothness of the Order

A random number

$$m \in [1, N]$$

is $N^{1/u}$-smooth with probability

$$u^{-u}.$$

* See Sutherland for details.

Minimize Expected Running Time

Assume the order of forms are like random integers.

Minimize Expected Running Time

Assume the order of forms are like random integers.

- Group operations per try is $O(N^{1/2r})$

Minimize Expected Running Time

Assume the order of forms are like random integers.

- Group operations per try is $O(N^{1/2r})$
- Assume order is random $h \in [1, N^{1/2}]$

Minimize Expected Running Time

Assume the order of forms are like random integers.

- Group operations per try is $O(N^{1/2r})$
- Assume order is random $h \in [1, N^{1/2}]$
- Probability of factoring is r^{-r}

Minimize Expected Running Time

Assume the order of forms are like random integers.

- Group operations per try is $O(N^{1/2r})$
- Assume order is random $h \in [1, N^{1/2}]$
- Probability of factoring is r^{-r}
- Expectation is $O(r^r N^{1/2r})$

Minimize Expected Running Time

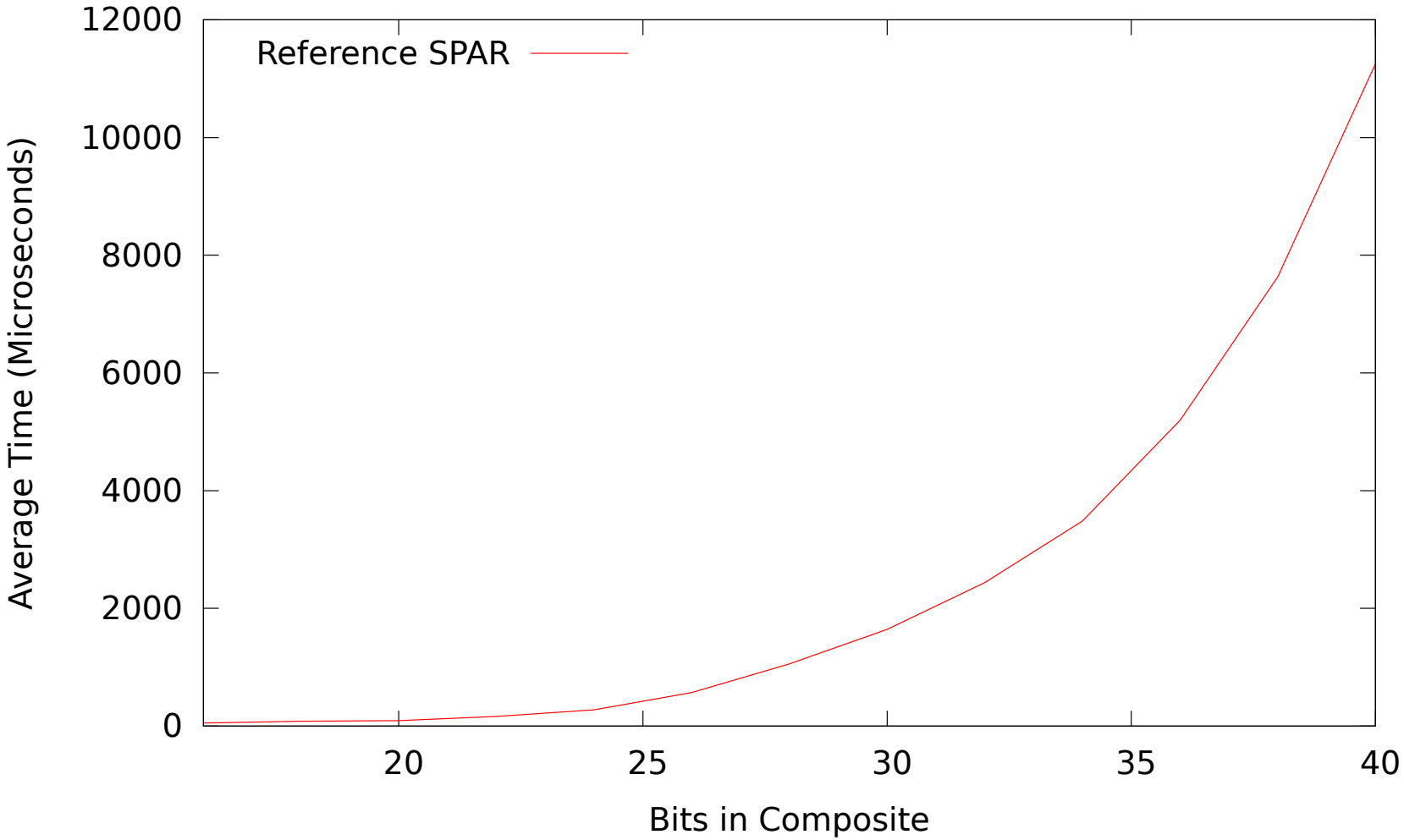
Assume the order of forms are like random integers.

- Group operations per try is $O(N^{1/2r})$
- Assume order is random $h \in [1, N^{1/2}]$
- Probability of factoring is r^{-r}
- Expectation is $O(r^r N^{1/2r})$
- Minimized for $r = \sqrt{\ln N / \ln \ln N}$

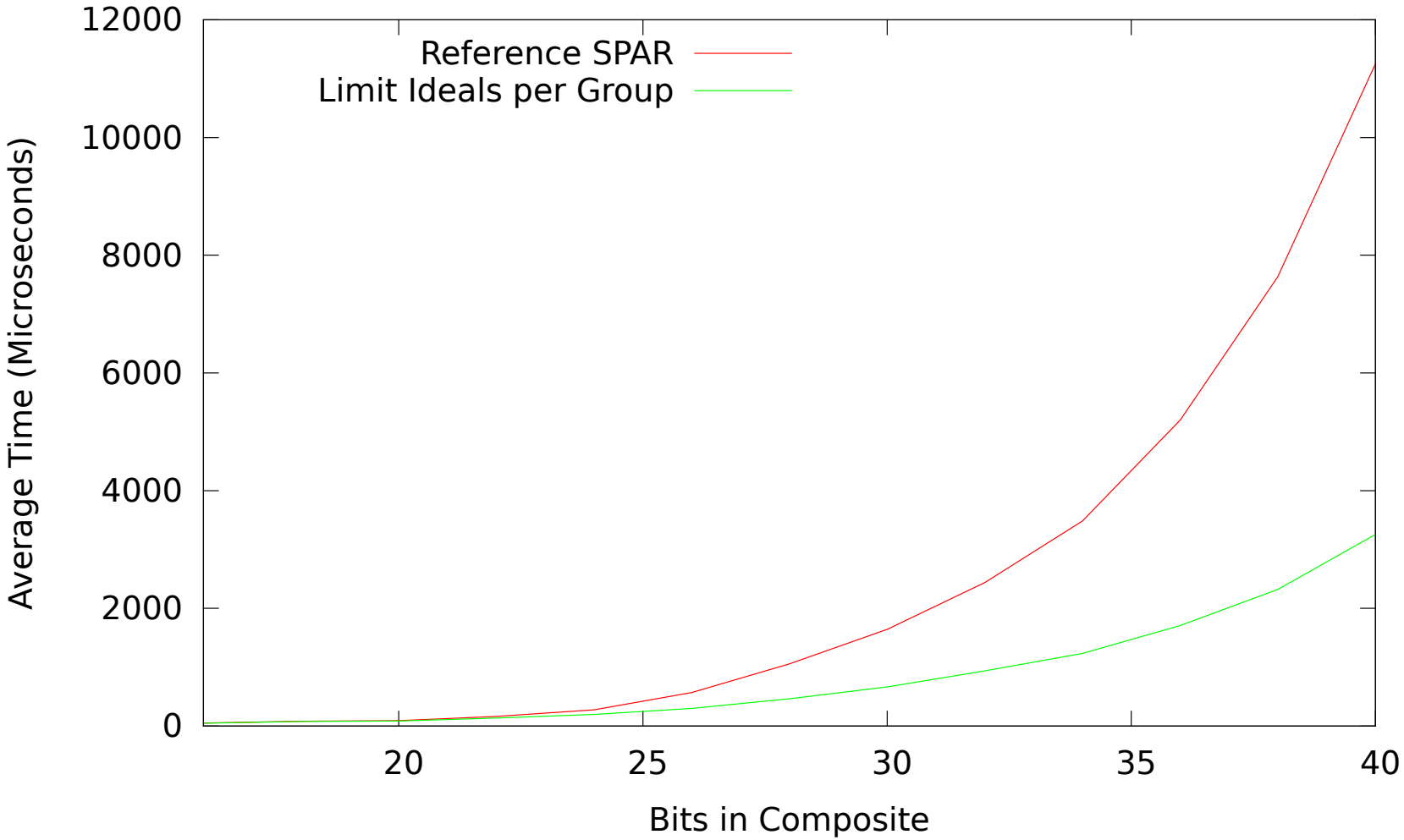
How to Handle Failure?

- Try again with a different form.
- Try again with a different $\Delta = -kN$.

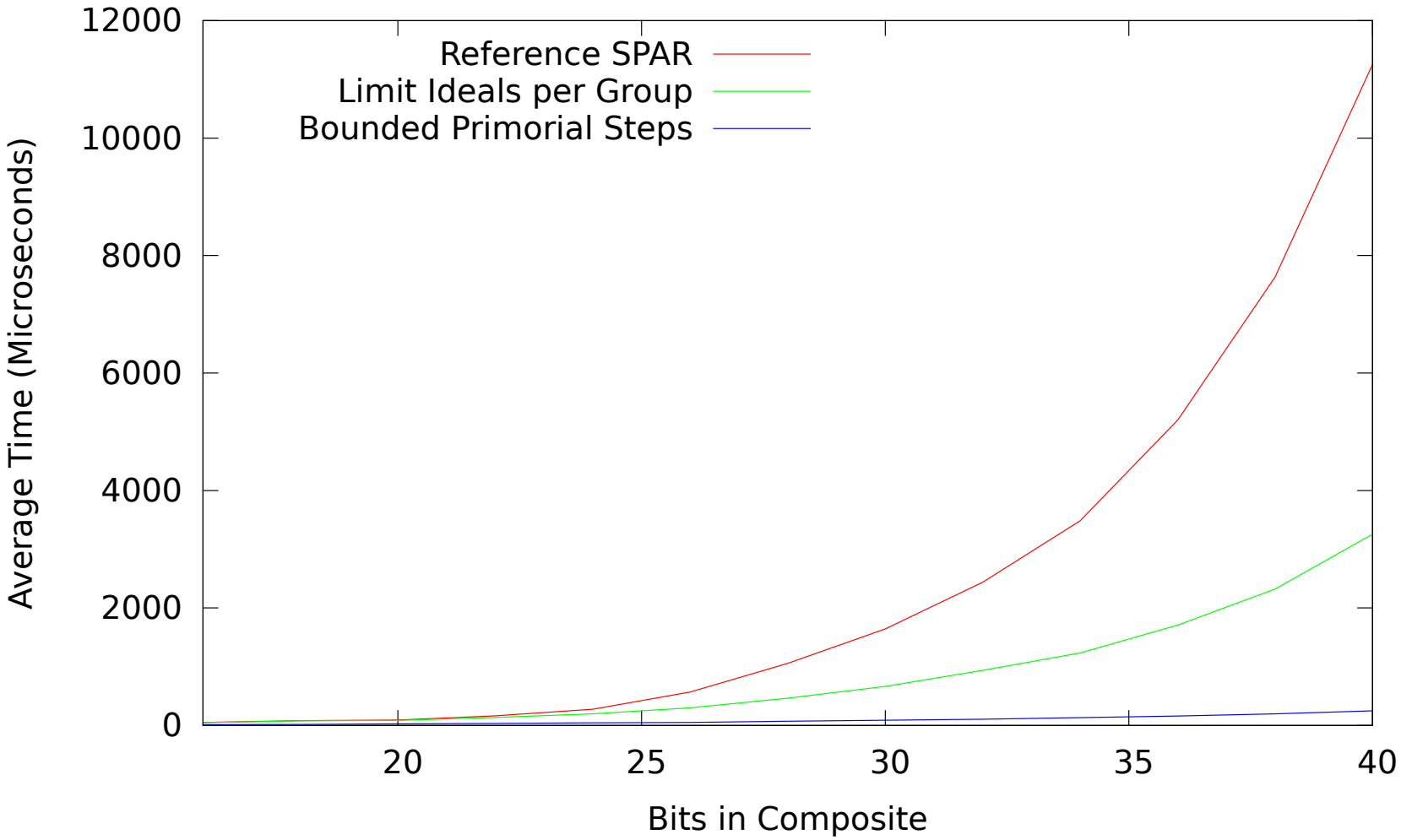
Reference Implementation of SPAR



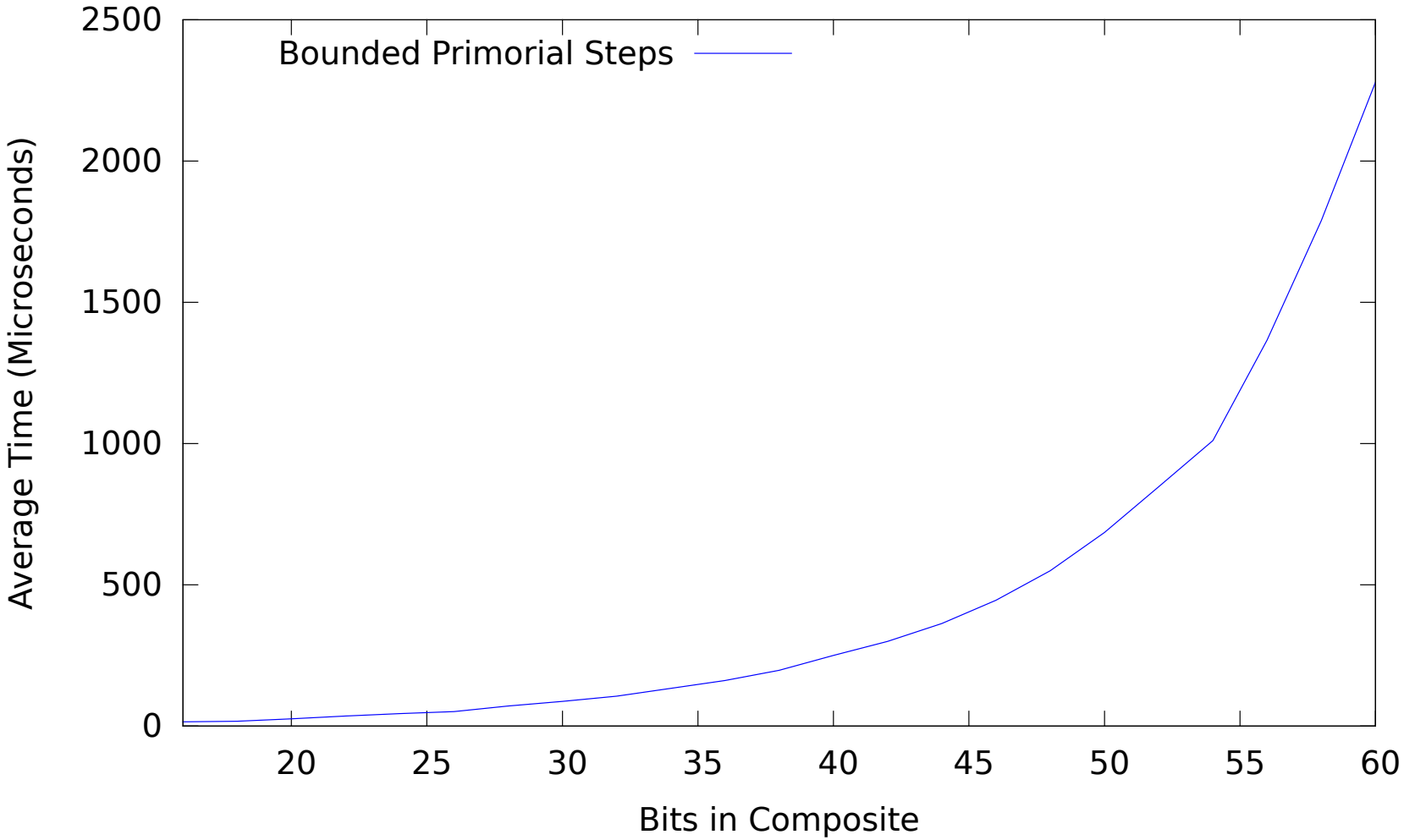
Bound the Number of Ideals per Group



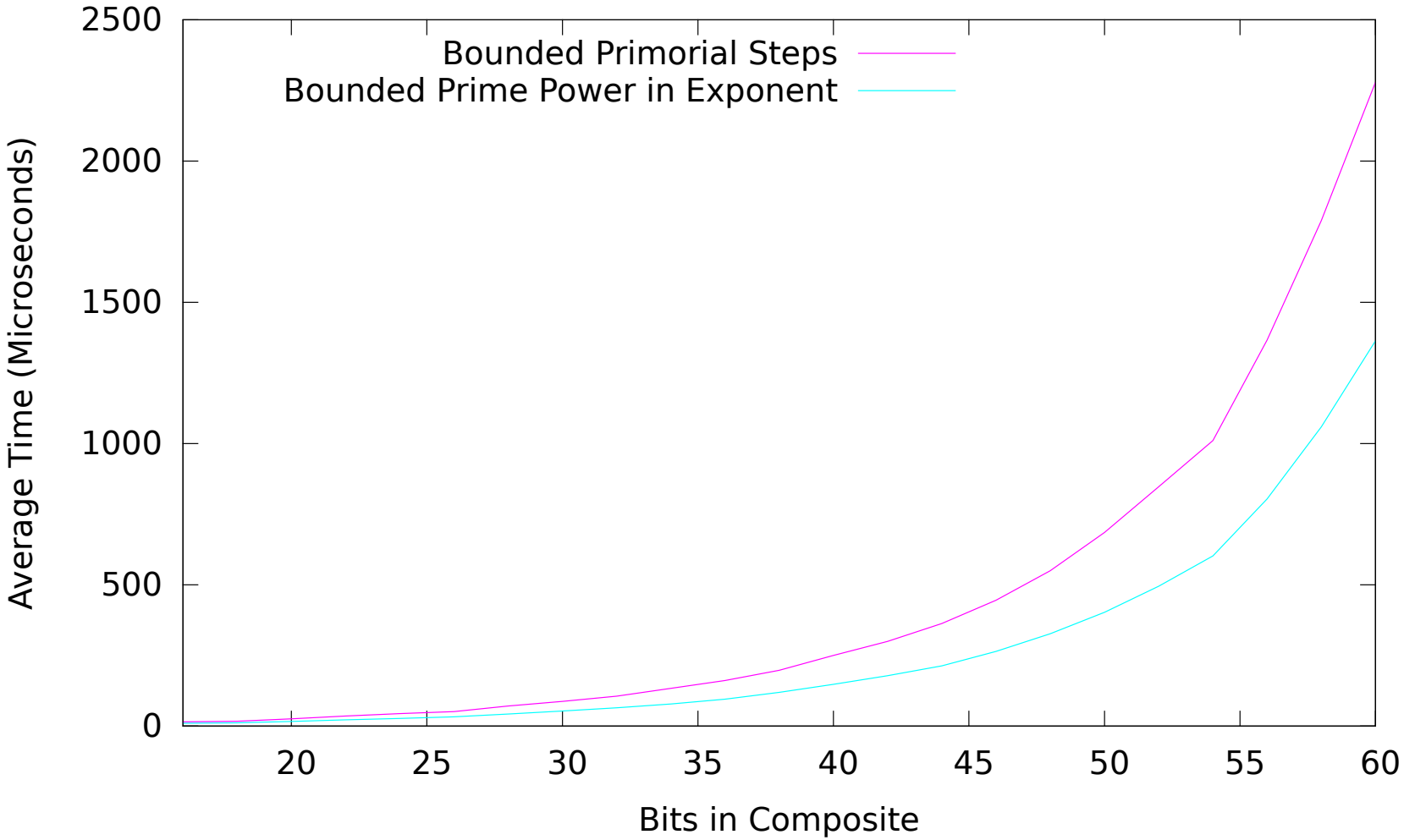
Bounded Primorial Steps Search



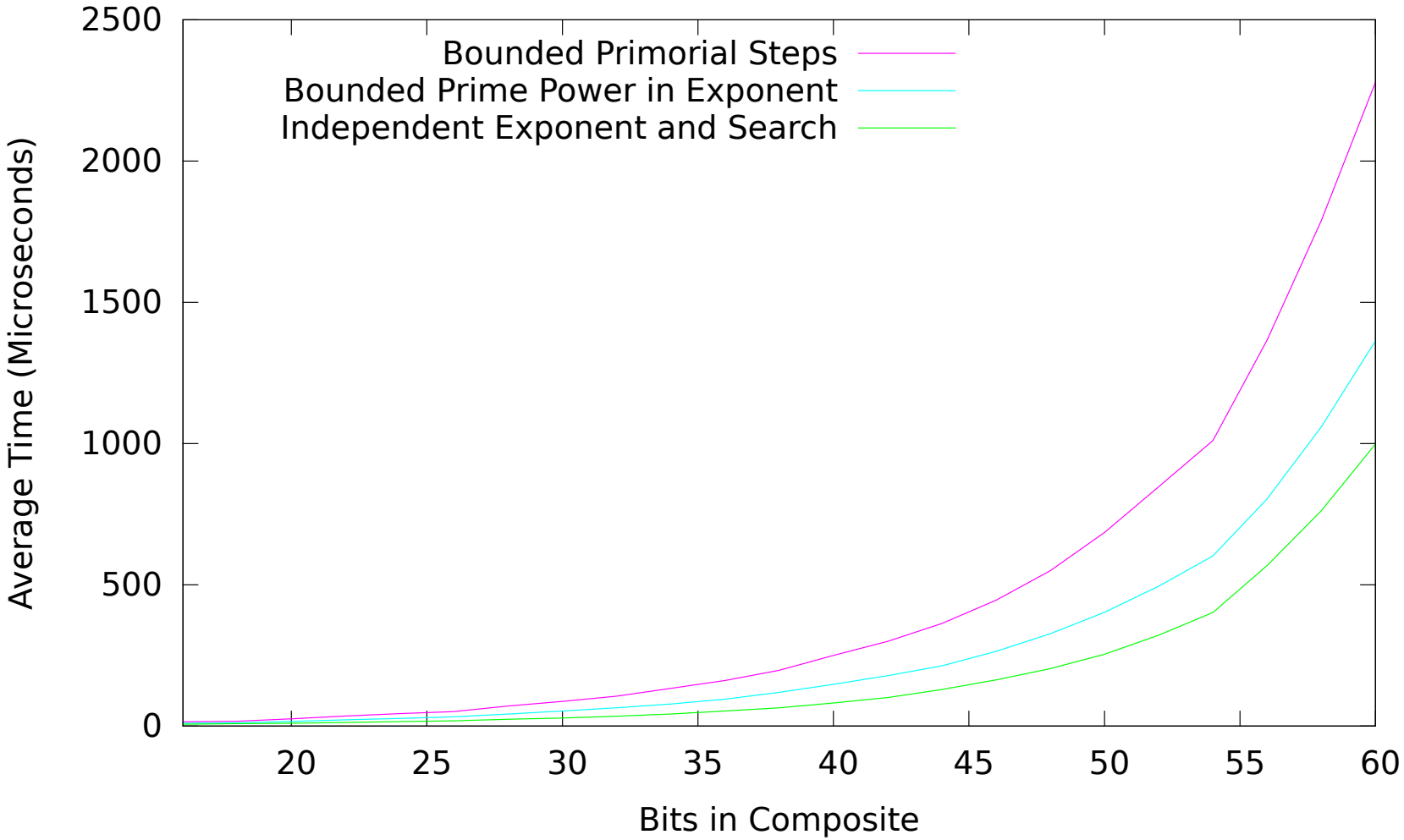
Bounded Primorial Steps Search



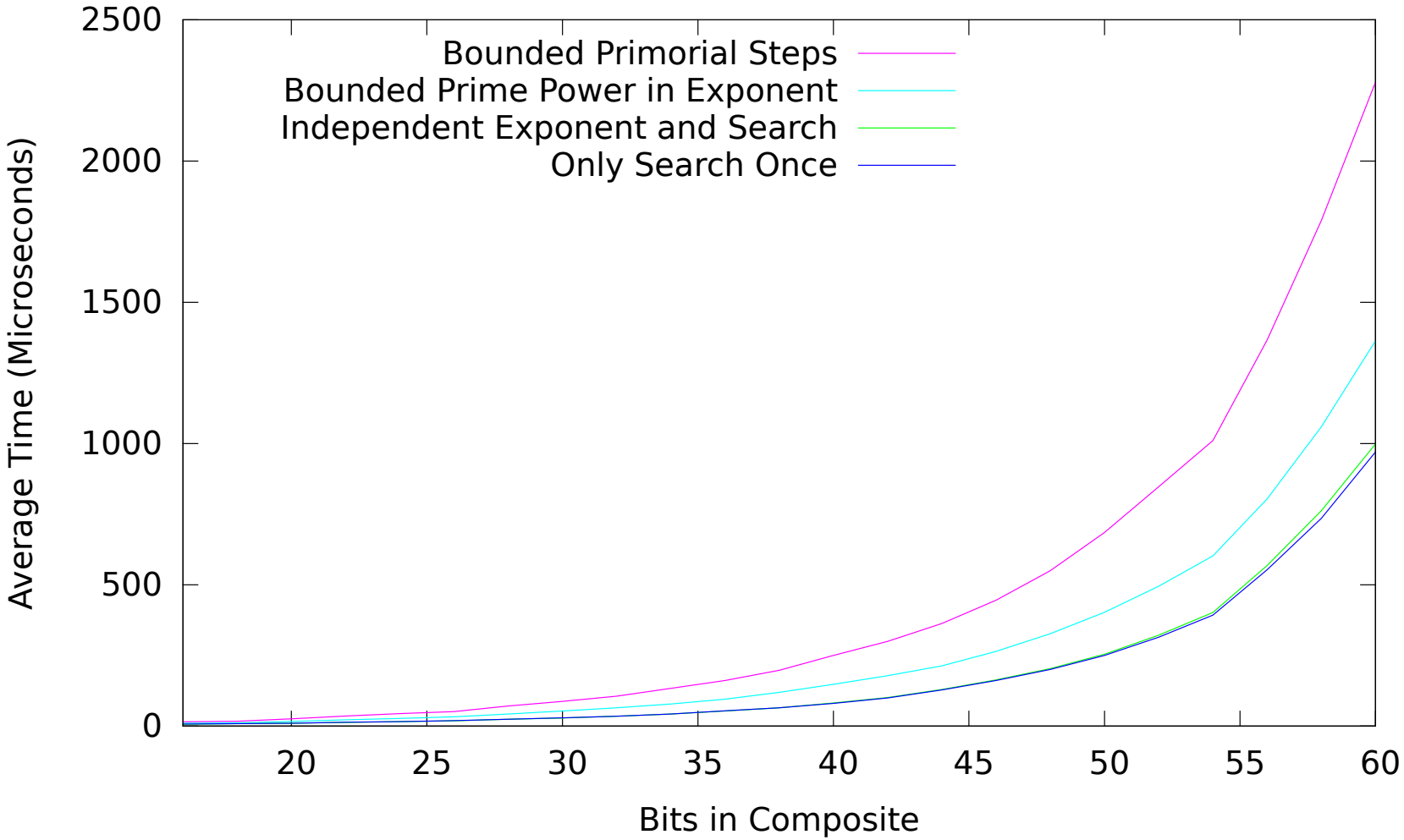
Bound e_i in Exponent $E = \prod {p_i}^{e_i}$



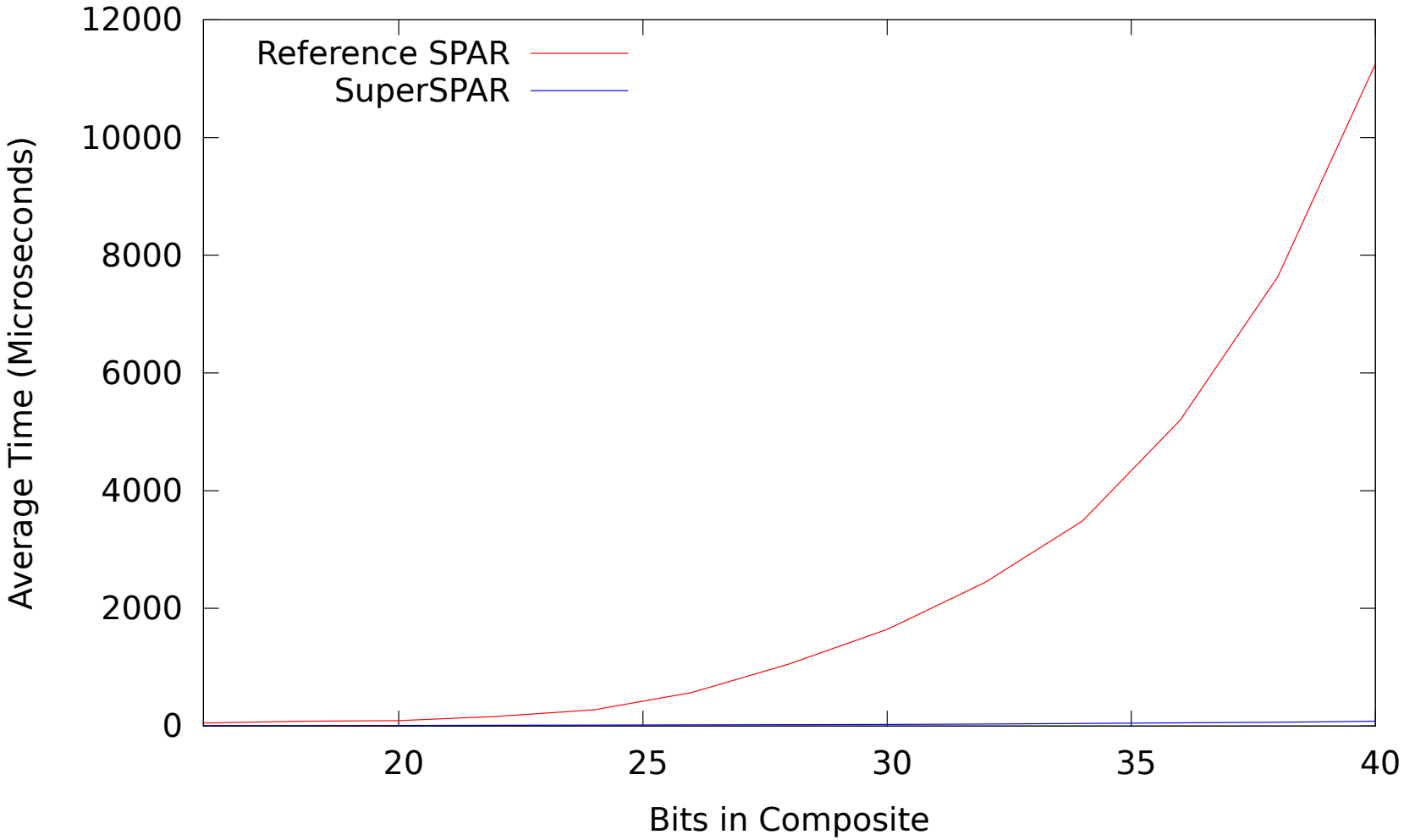
Independent Exponent E and Search Bound mP_w



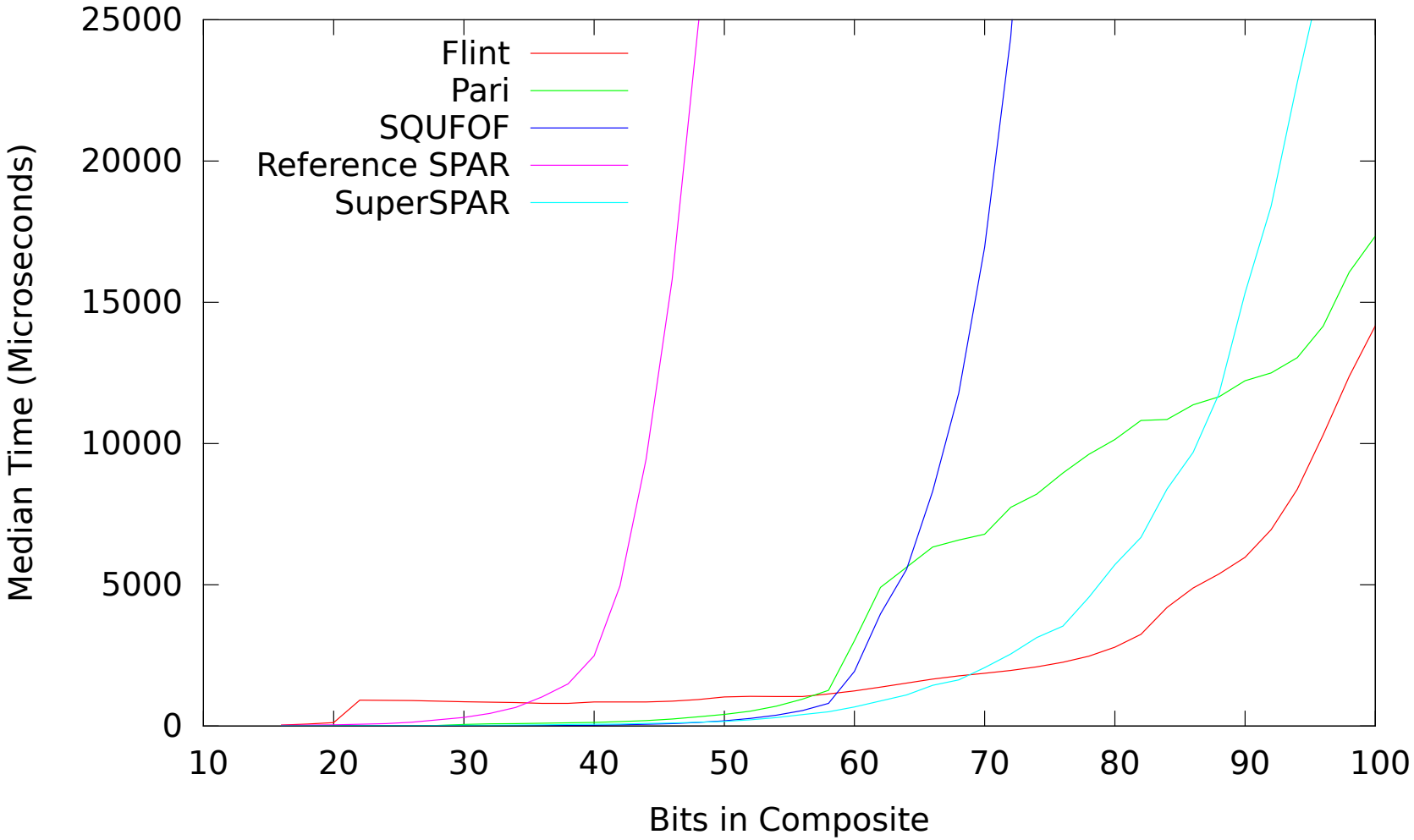
Perform a Single Search



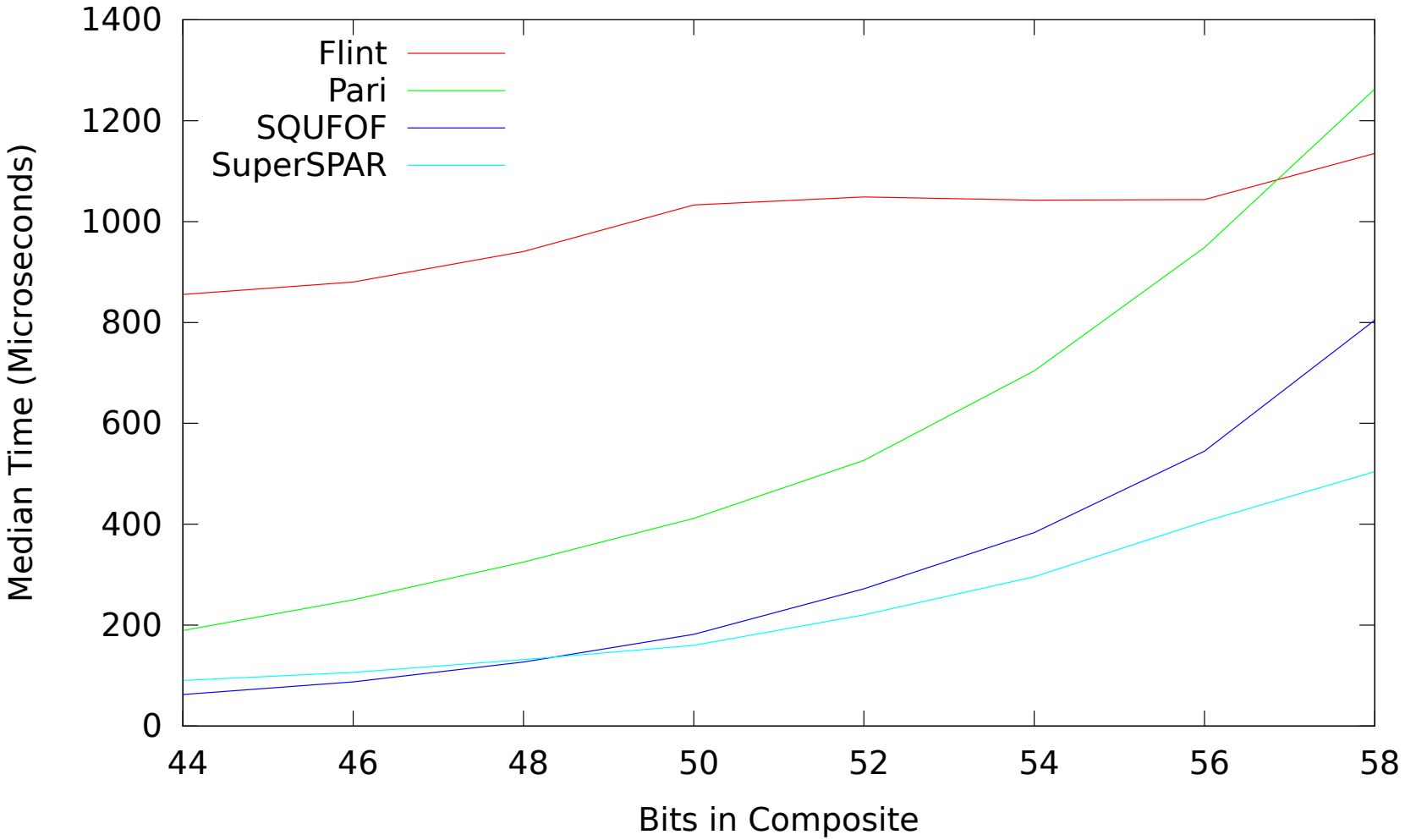
SPAR vs SuperSPAR



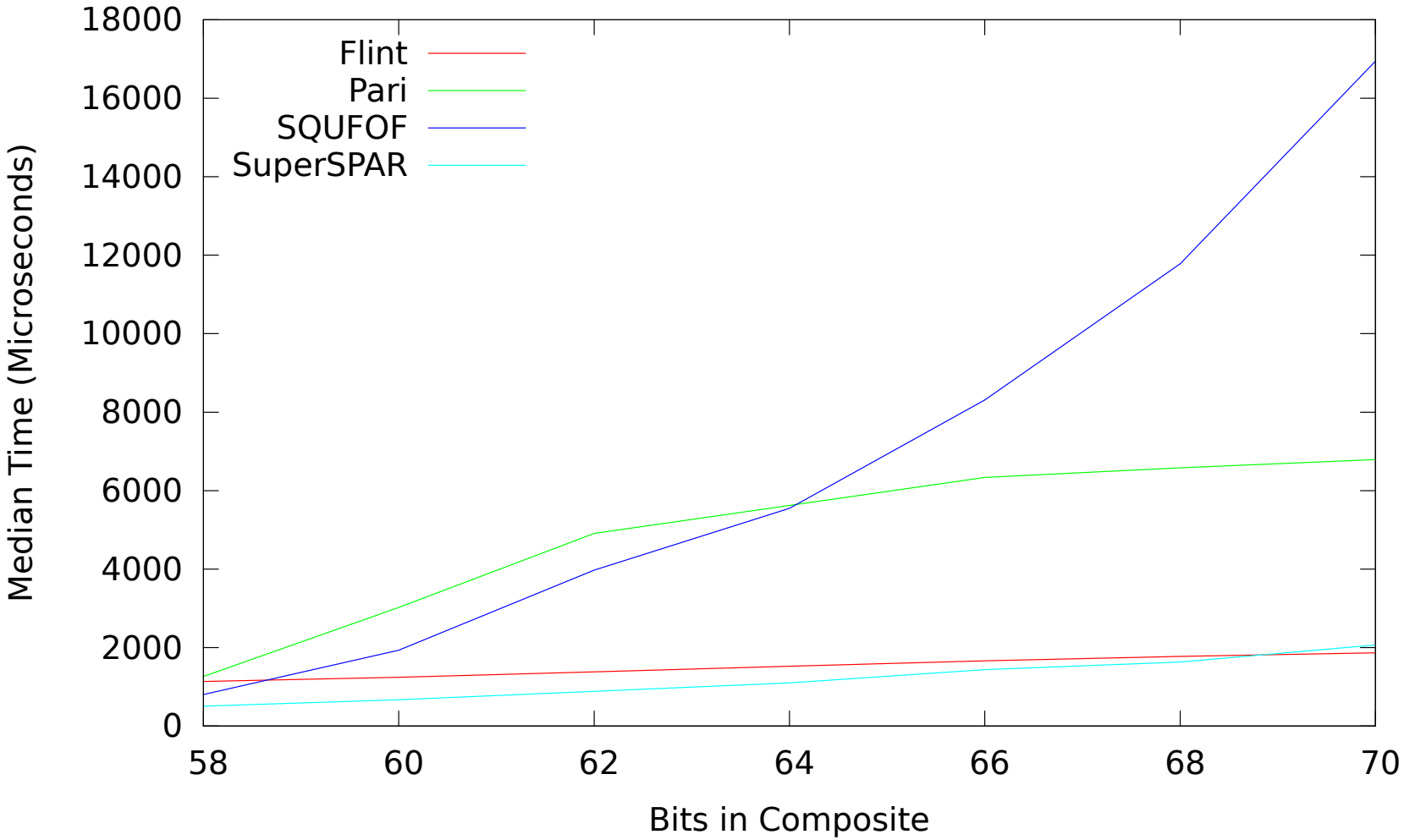
Median Integer Factoring Times



Median Integer Factoring Times (Zoomed Left)



Median Integer Factoring Times (Zoomed Right)



Thank You

Questions?

Future Work

Some work that is eagerly awaiting:

- Special case for quotients $q \leq 4$ in XGCD.
- Real quadratic fields and hyperelliptic curves.
- Improvements to left-to-right best approximations algorithm.

Some Possible Applications

These are just a few possible applications.

Ideal class group:

- Class group tabulation for larger discriminants (Ramachandran2006).

Integer Factoring:

- Factoring left-overs after sieving in algorithms like the number field sieve using multiple large primes.

Contributions

Contributions to ideal class group arithmetic:

- Optimized implementations using 64 and 128-bit arithmetic.
- Optimized XGCD for 32, 64, and 128-bits.
  - A simplified left-to-right binary XGCD (and partial XGCD).

Contributions to integer factoring:

- SuperSPAR integer factoring algorithm.
- Left-to-right best approximations method for computing 2,3 representations of power primorials.

Extended Greatest Common Divisor (XGCD)

Extended greatest common divisor solves

$$s = Ua + Vb = \gcd(a, b)$$

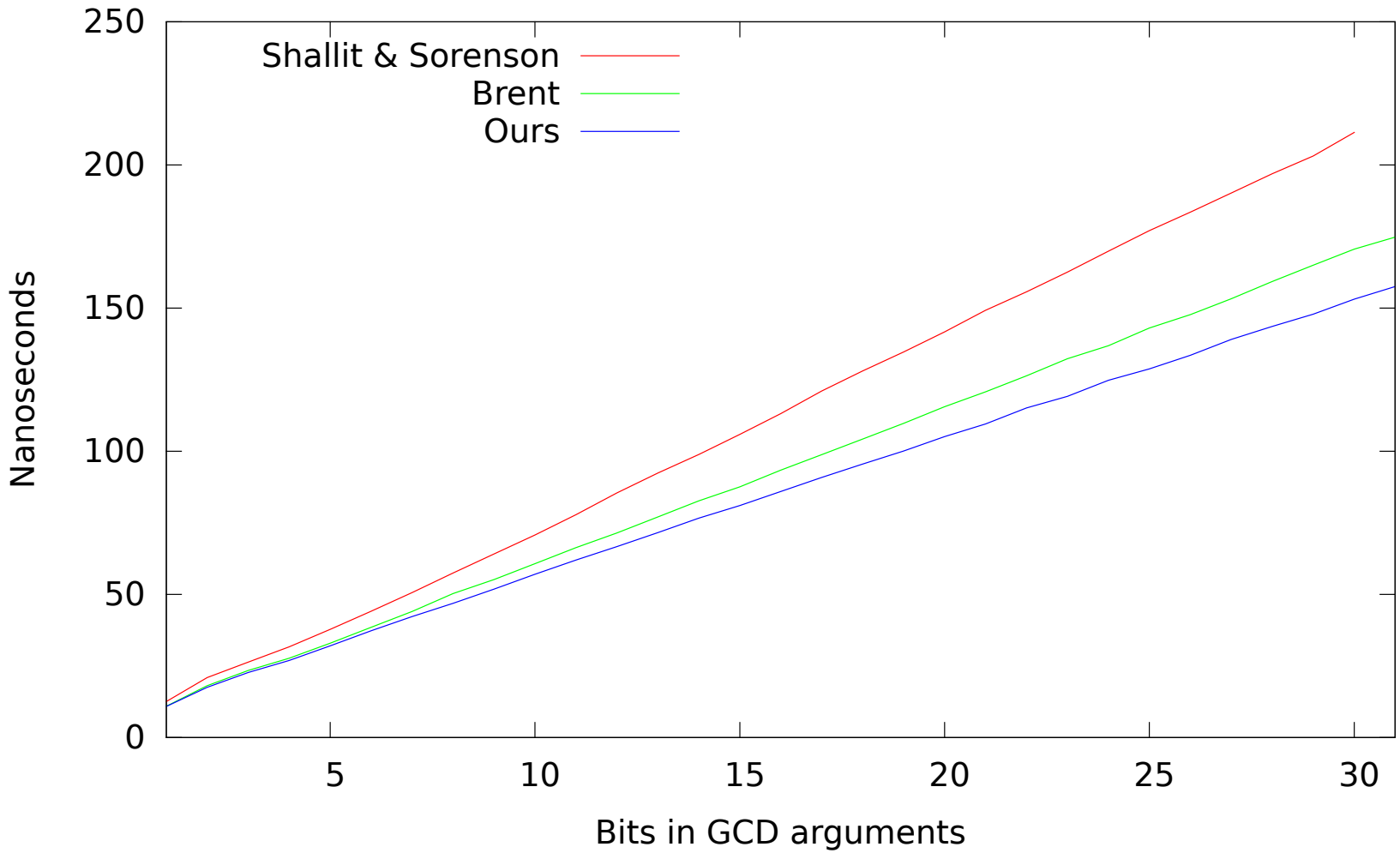
for the largest s dividing both a and b.

Our Left-to-Right Binary PXGCD

1: **procedure** OUR_L2R_PXGCD(a, b, T) $\quad \{a, b, T \in \mathbb{Z}_{\geq 0}\}$
2: $\quad \begin{bmatrix} r_1 & c_1 \\ r_0 & c_0 \end{bmatrix} \leftarrow \begin{bmatrix} a & 0 \\ b & -1 \end{bmatrix}$
3: $\quad z \leftarrow 0, s_1 \leftarrow 1, s_0 \leftarrow 1$
4: $\quad$ **while** $r_0 \neq 0$ and $r_0 > T$ **do**
5: $\quad\quad k \leftarrow$ NUMBITS(r_1) - NUMBITS(r_0)
6: $\quad\quad$ subtract 2^k times the second row from the first
7: $\quad\quad$ **if** $r_1 < 0$ **then** $r_1 \leftarrow -r_1, c_1 \leftarrow -c_1, s_1 \leftarrow -s_1$
8: $\quad\quad$ **if** $r_1 < r_0$ **then**
9: $\quad\quad\quad$ swap the rows of the matrix and s_1 with s_0
10: $\quad\quad\quad z \leftarrow z + 1$
11: $\quad$ **return** $(s_1 \cdot r_1, s_1 \cdot c_1, s_0 \cdot r_0, s_0 \cdot c_0, z)$

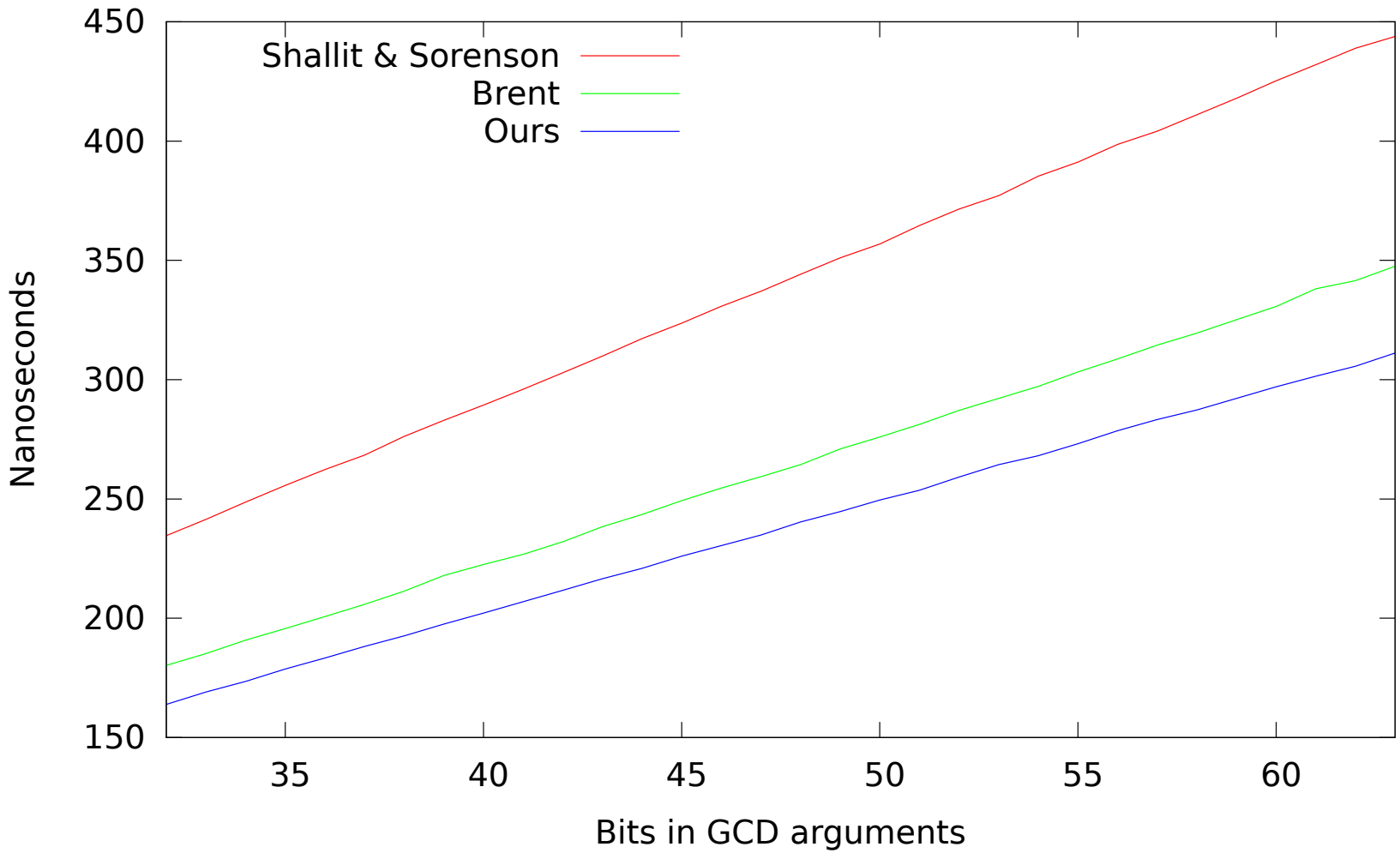
Left-to-Right Binary XGCDs

32-bit



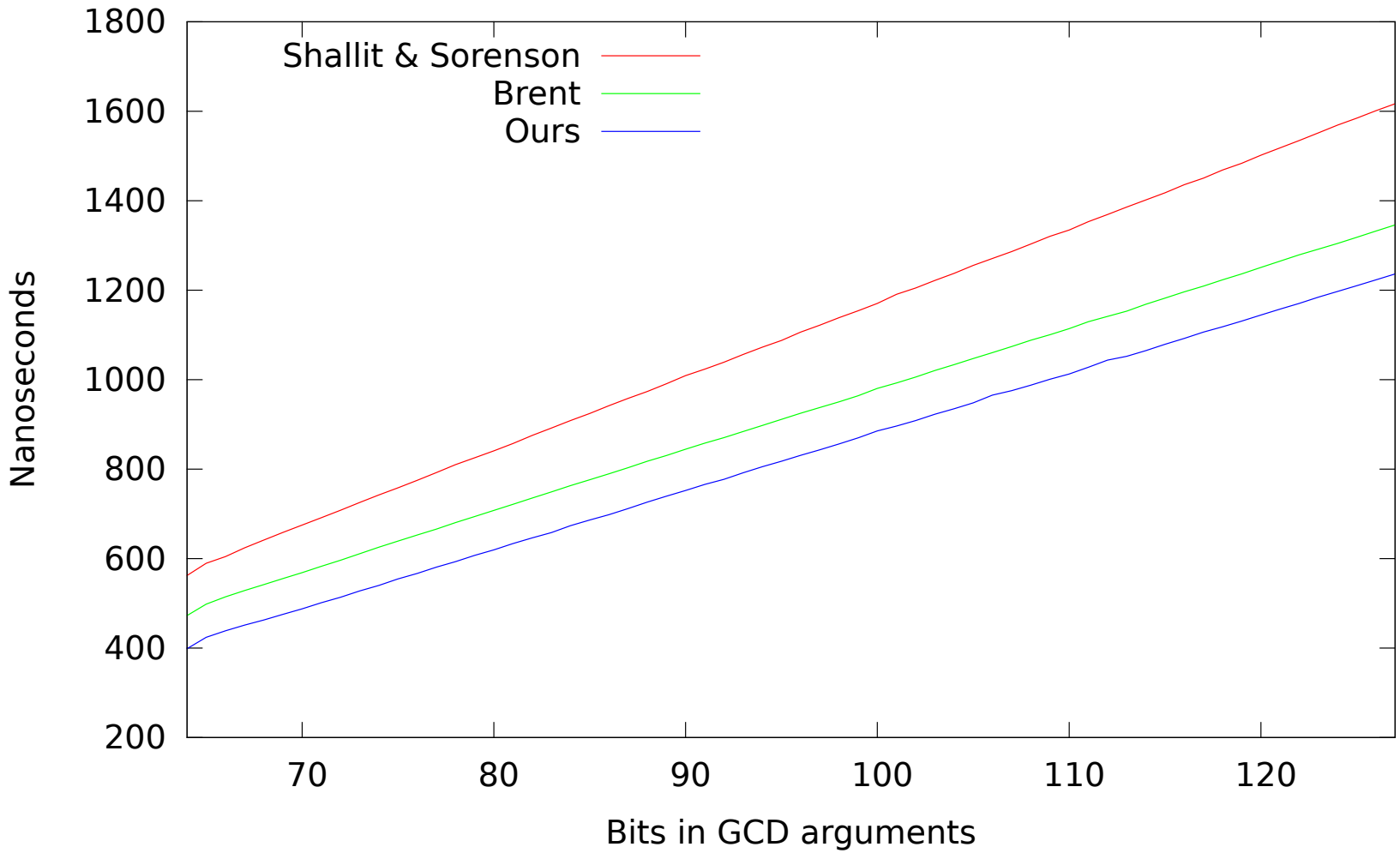
Left-to-Right Binary XGCDs

64-bit



Left-to-Right Binary XGCDs

128-bit



Approximate Square Root

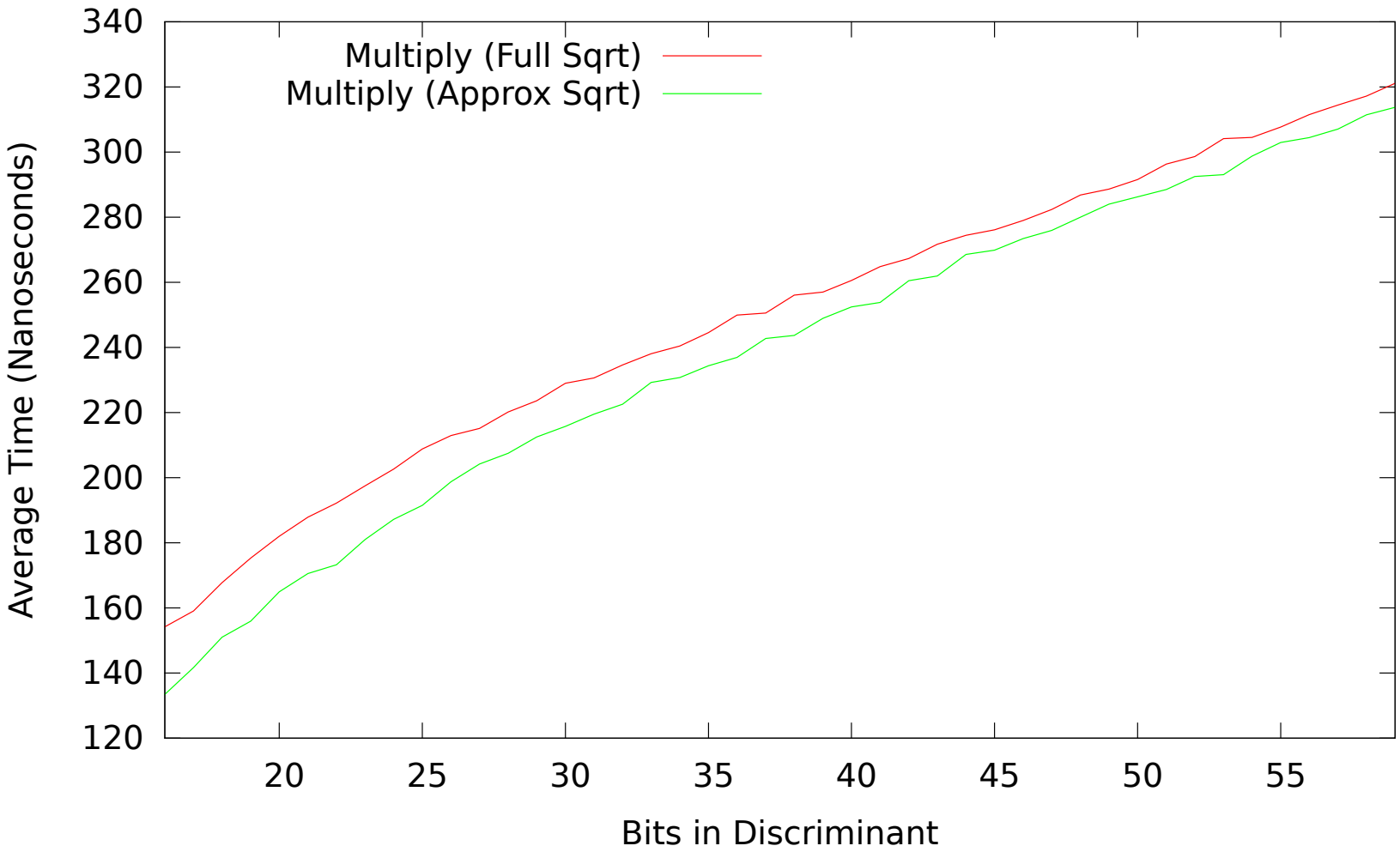
Notice that

$$\begin{aligned} x^{1/2} &= 2^{(\log_2 x)/2} &&\approx 2^{\lfloor \lfloor \log_2 x + 1 \rfloor / 2 \rfloor} \\ \Rightarrow \quad x/x^{1/2} &= x/2^{(\log_2 x)/2} &&\approx x/2^{\lfloor \lfloor \log_2 x + 1 \rfloor / 2 \rfloor}, \end{aligned}$$

which is approximated by shifting x right by $\lfloor \lfloor \log_2 x + 1 \rfloor / 2 \rfloor$ bits.

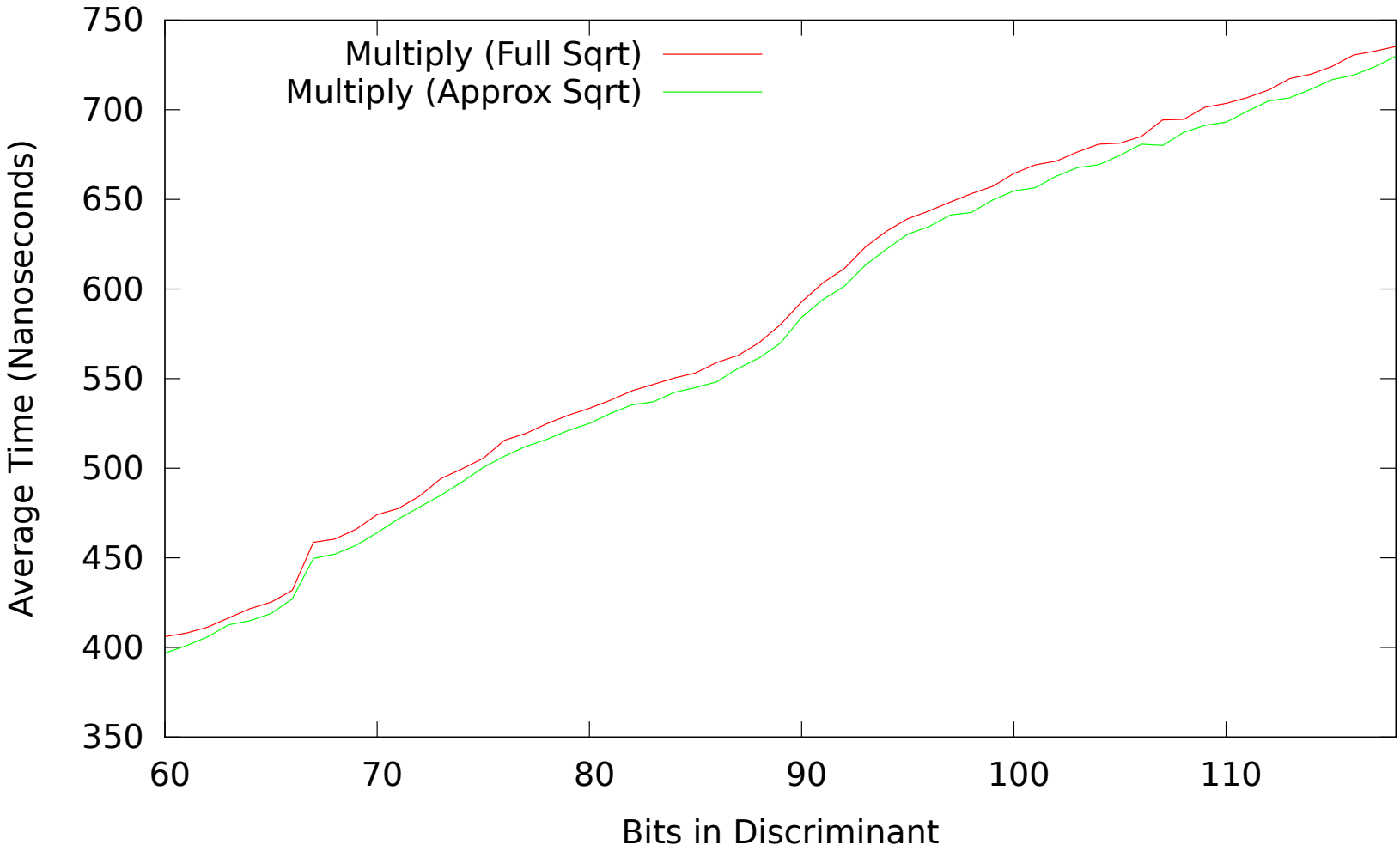
Square Root Approximation

64-bit ideal multiplication



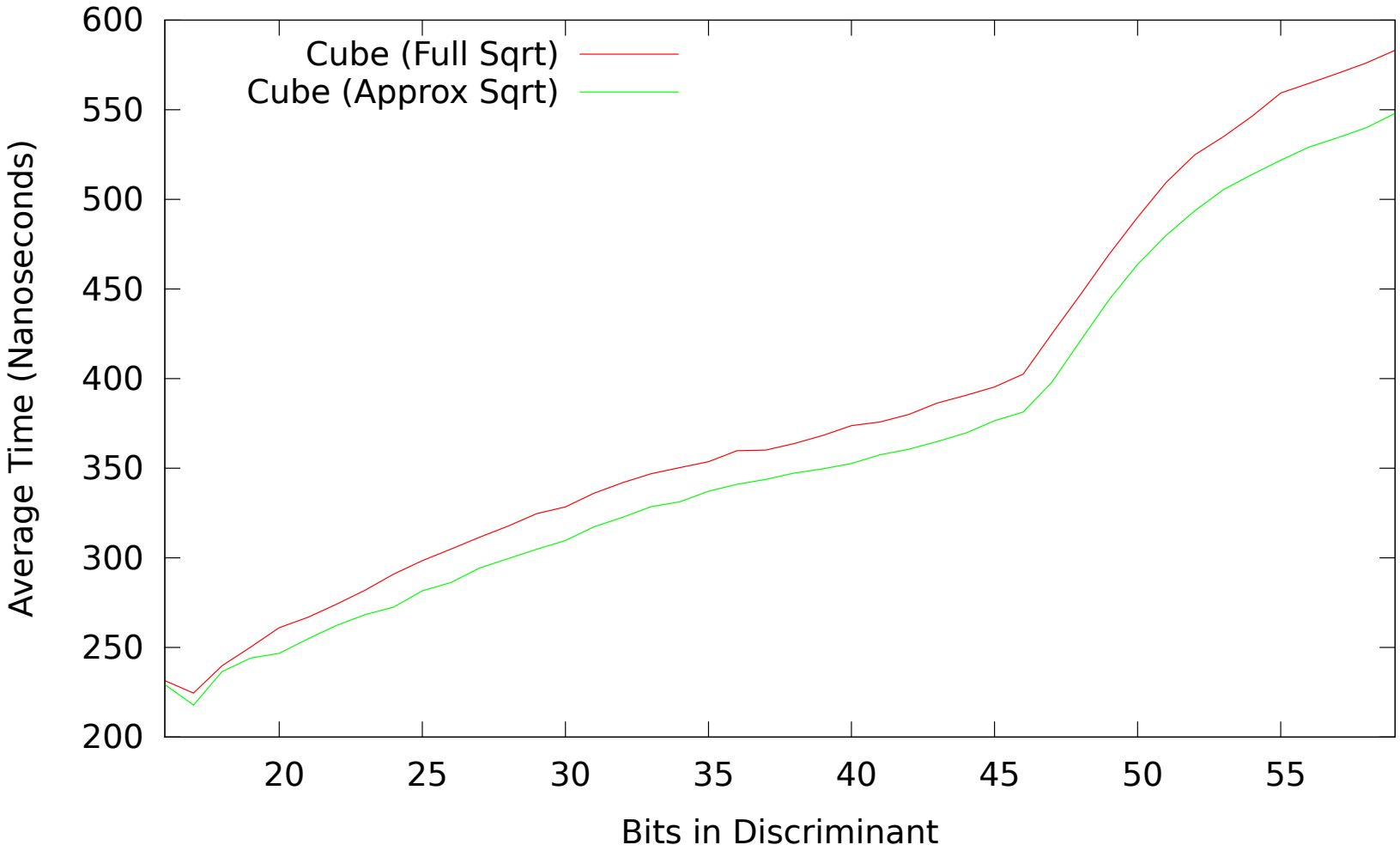
Square Root Approximation

128-bit ideal multiplication



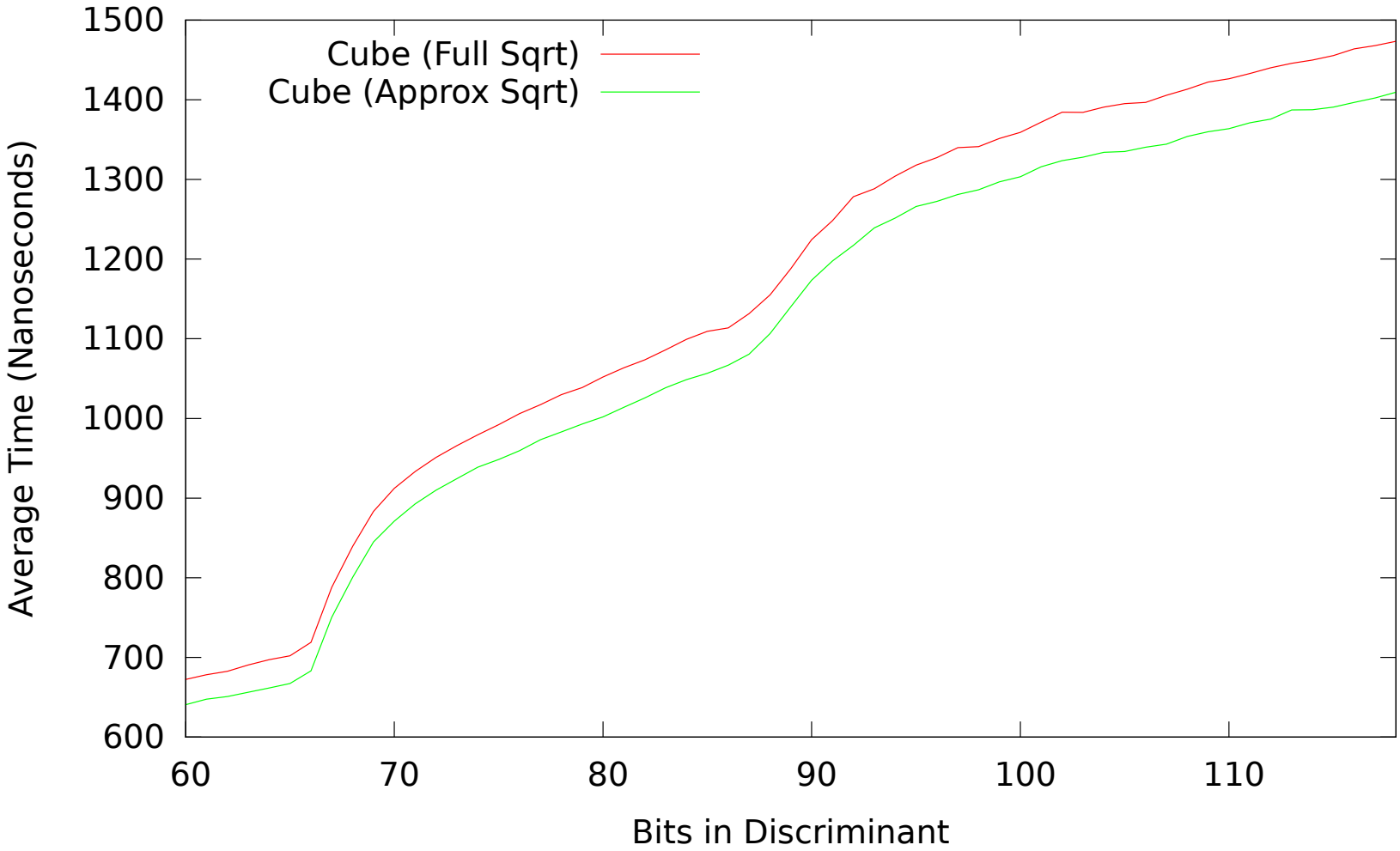
Square Root Approximation

64-bit ideal cubing



Square Root Approximation

128-bit ideal cubing



Large Intermediates During Cubing

Ideal Cubing (Algorithm 2.4, Page 19).

$$U = Y' c_1 (Y'(b_1 - Y' c_1 a_1) - 2) \bmod {a_1}^2 \qquad \{\text{line } 4\}$$

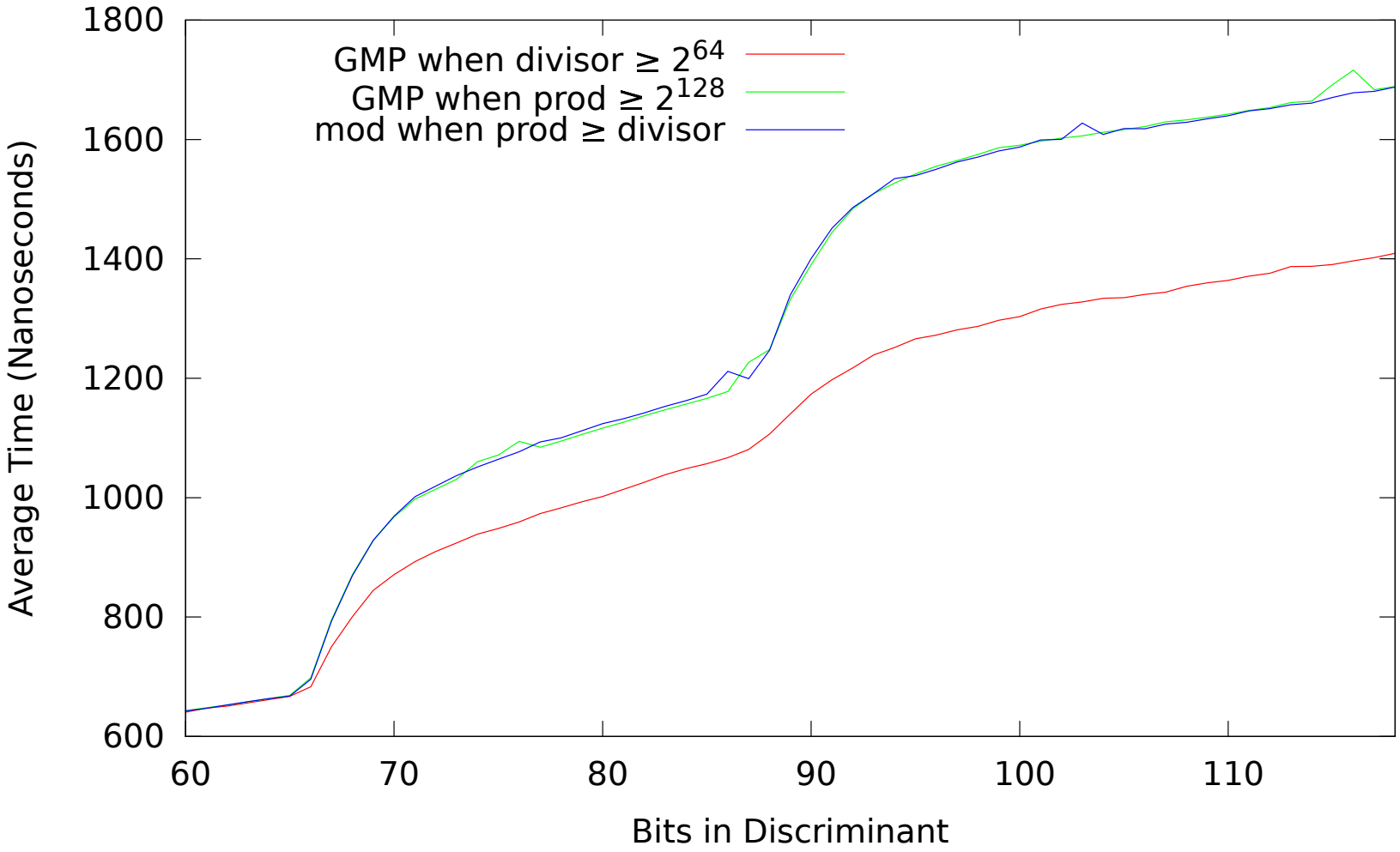
$$U = -c_1 (XY' a_1 + Y b_1) \bmod {a_1}^2 / s \qquad \{\text{line } 7\}$$

$$M_1 = \frac{(R_i a_1 + C_i U a_1)}{{a_1}^2} \qquad \{\text{line } 19\}$$

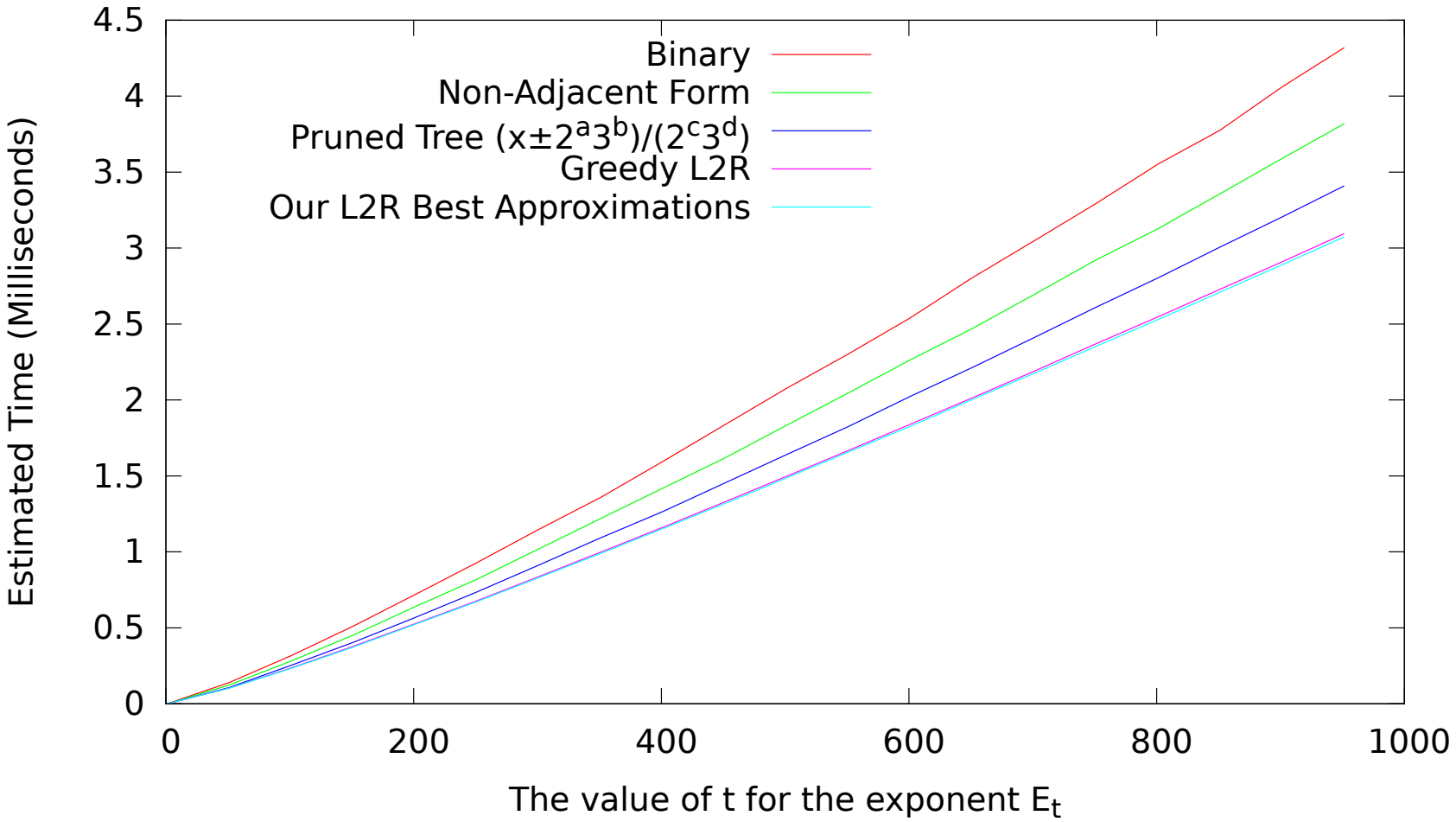
$$M_2 = \frac{R_i (b_1 + U a_1) - s C_i c_1}{{a_1}^2} \qquad \{\text{line } 20\}$$

The problem is $a_1 \leq \sqrt{|\Delta|/3}$ so ${a_1}^2 \leq |\Delta|/3$.

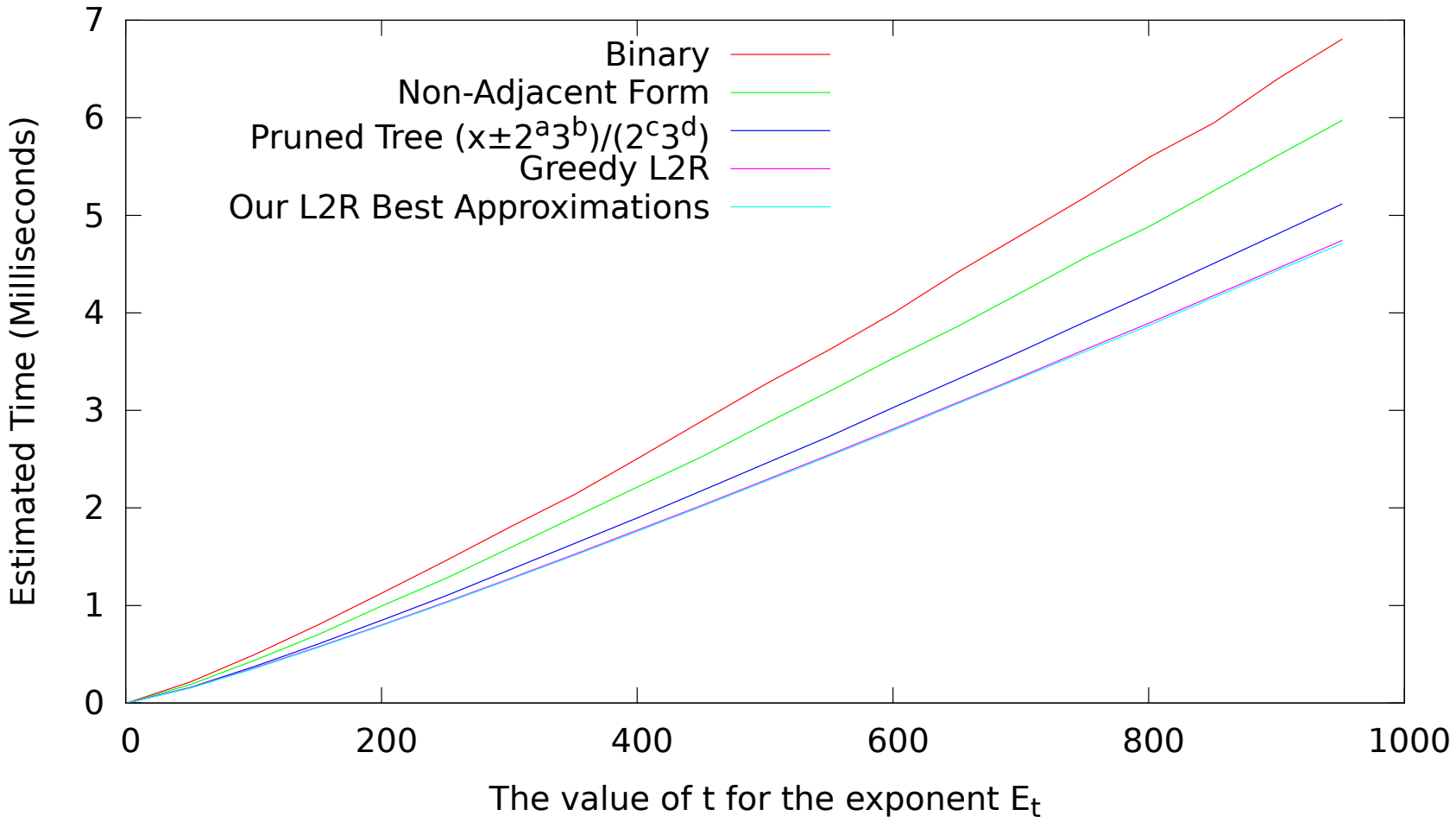
GMP with 128-bit Cubing



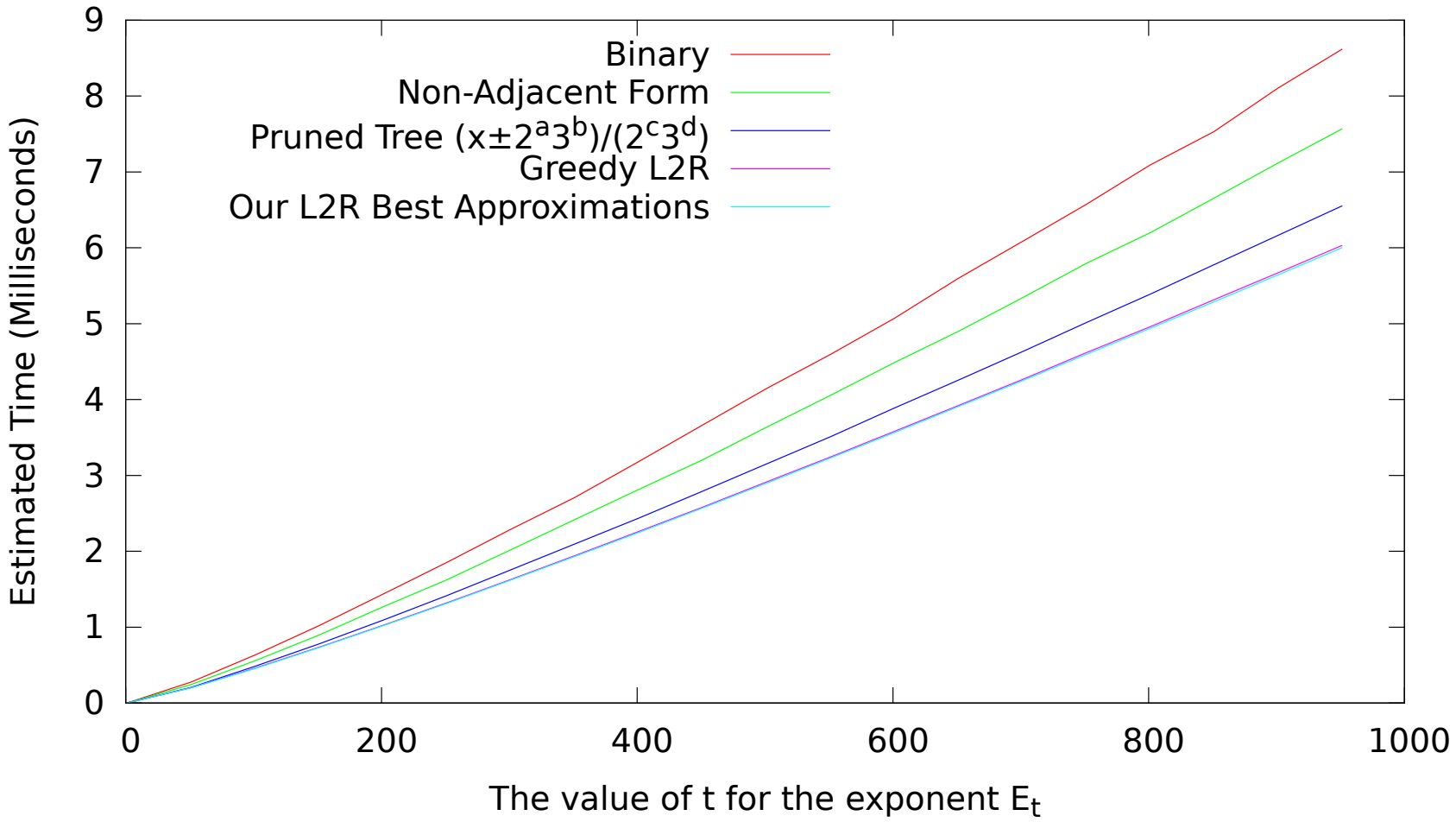
Exponentiation Results (16-bit Discriminants)



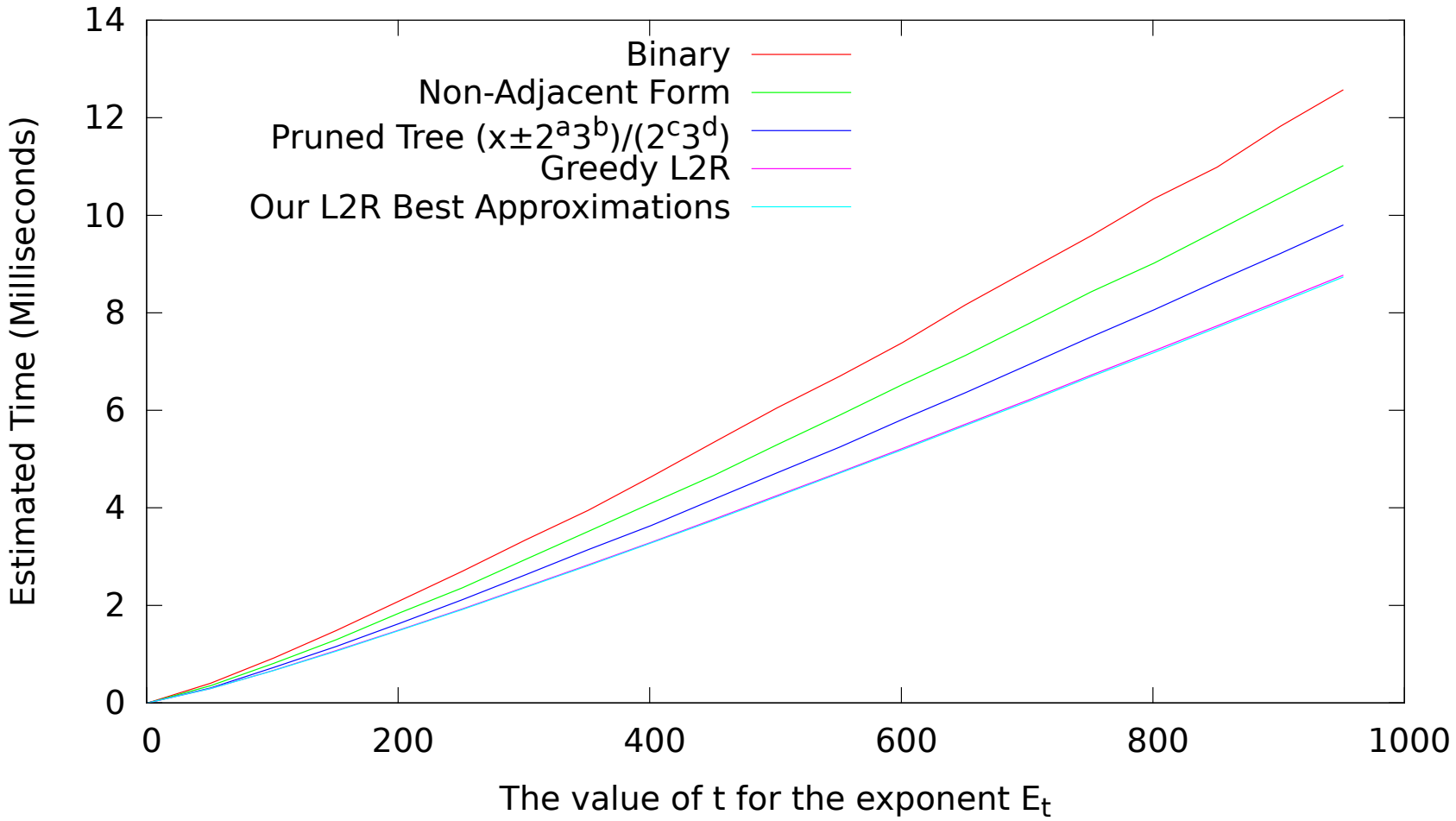
Exponentiation Results (32-bit Discriminants)



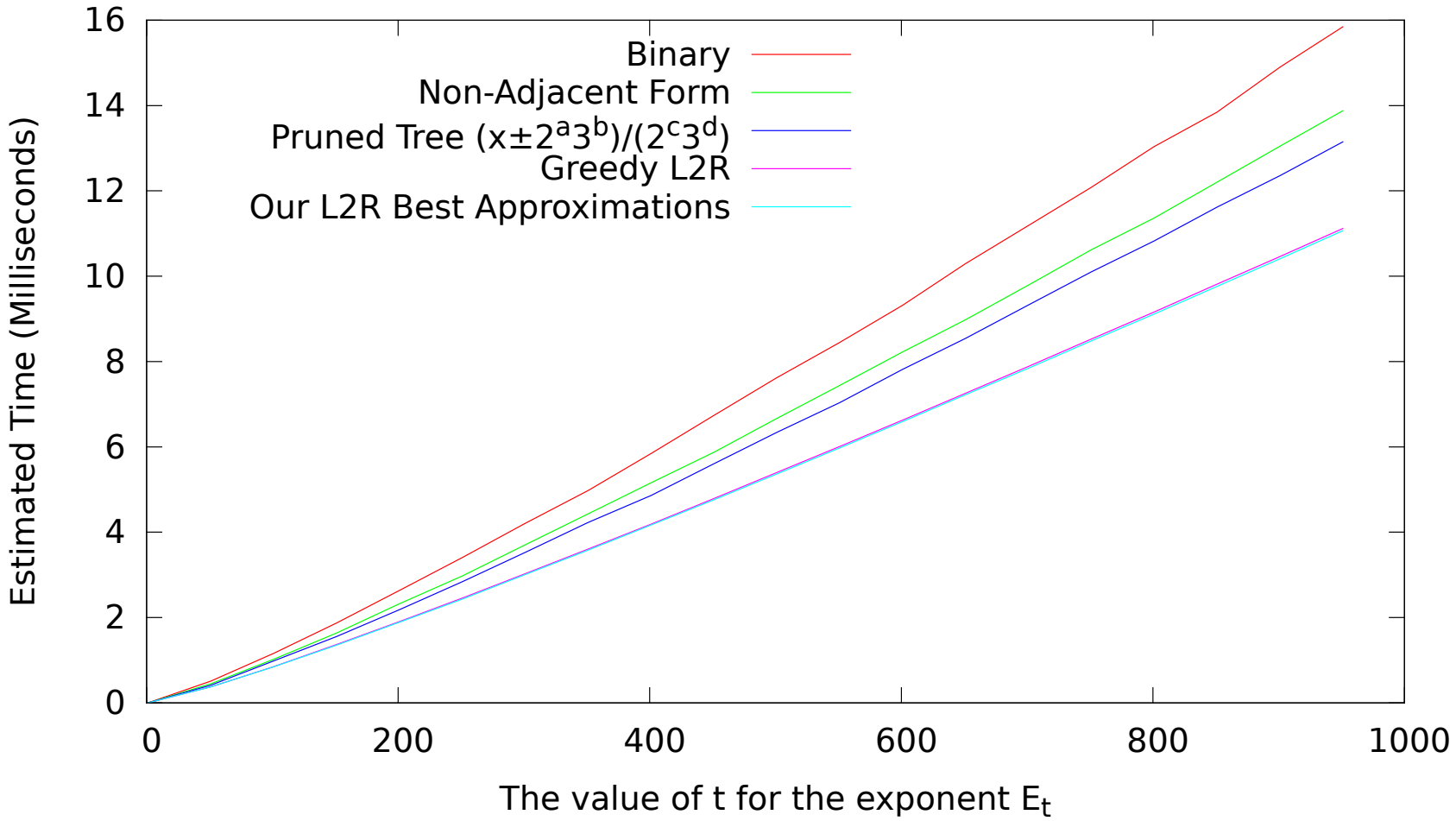
Exponentiation Results (48-bit Discriminants)



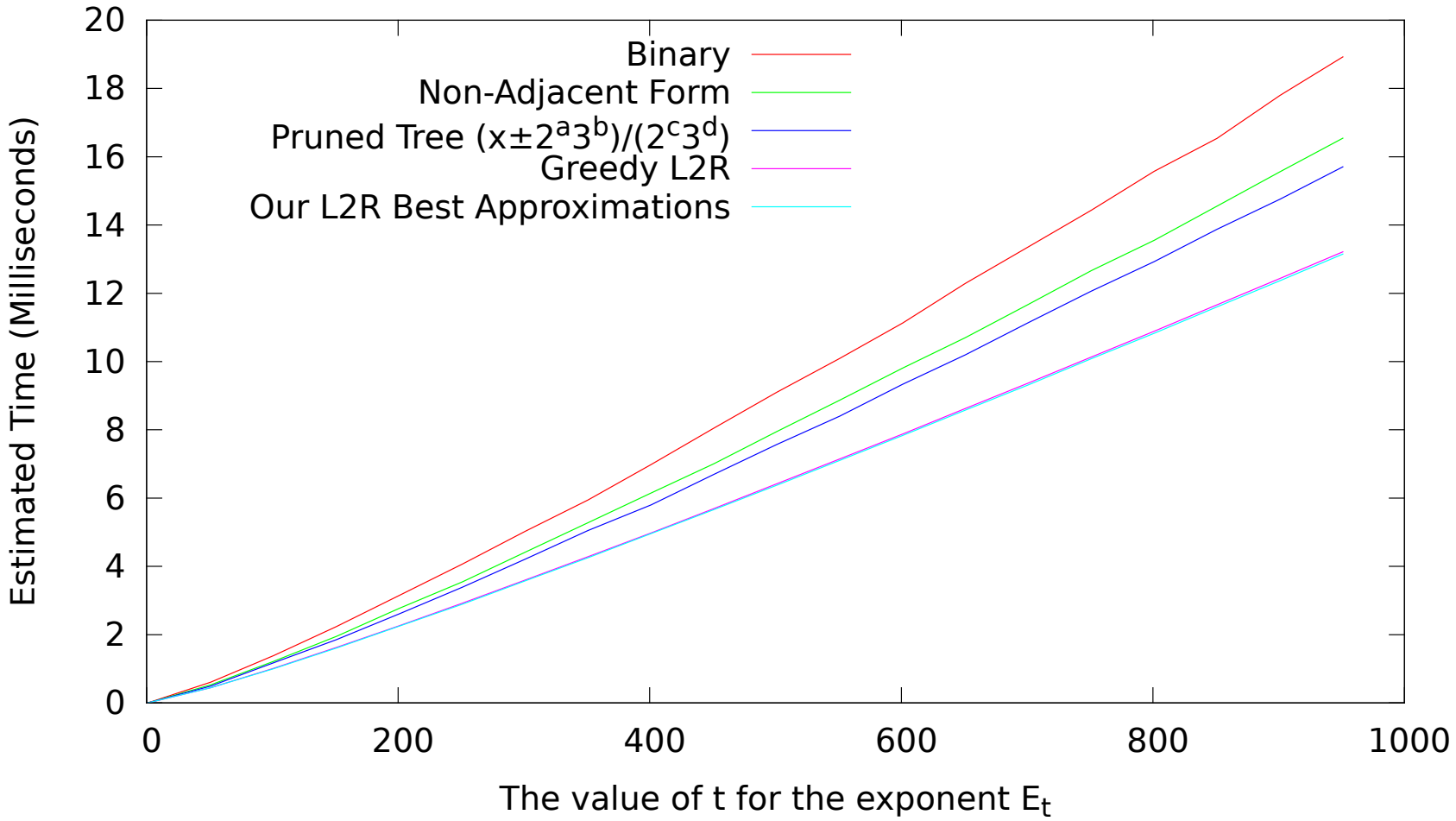
Exponentiation Results (64-bit Discriminants)



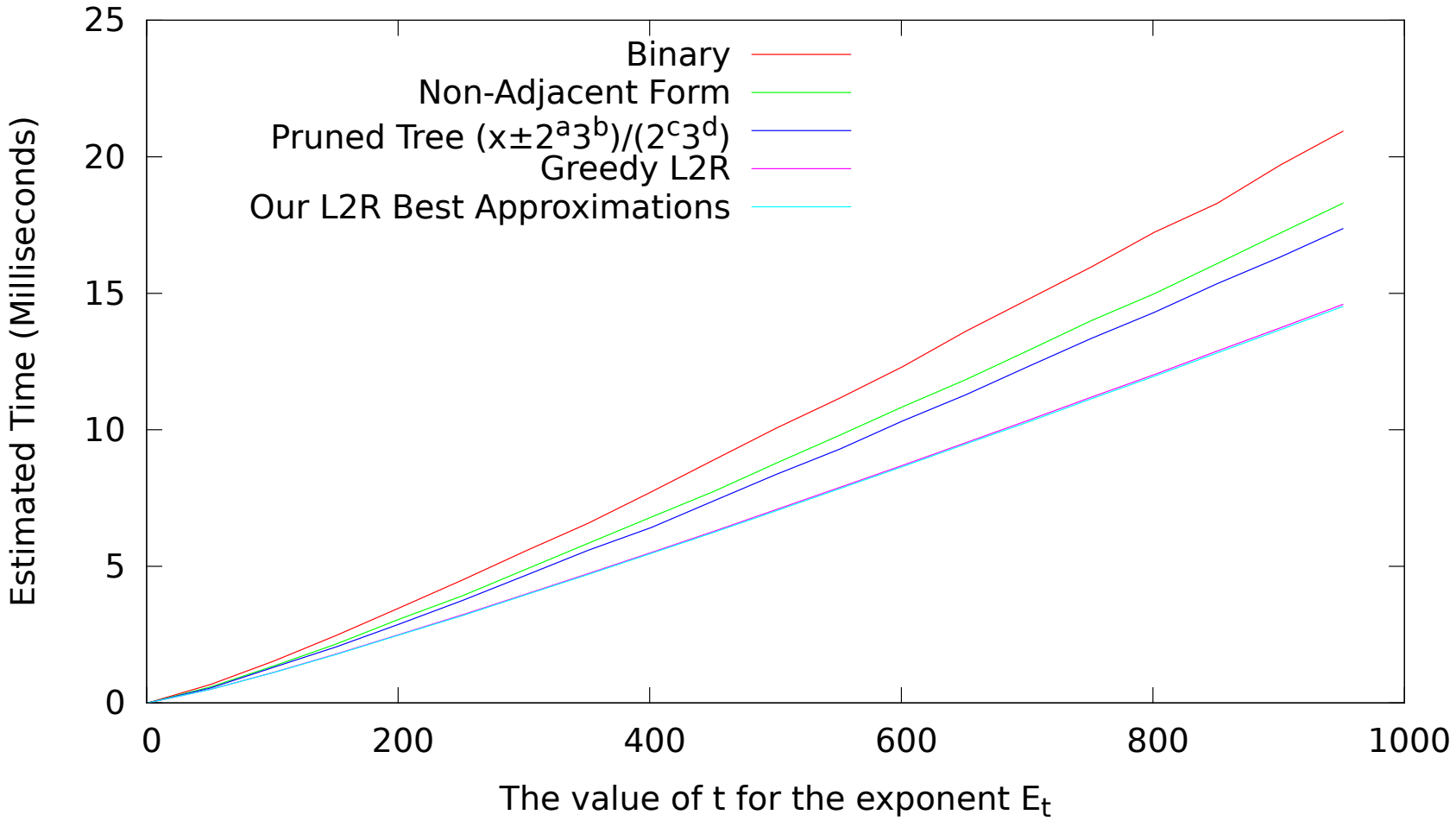
Exponentiation Results (80-bit Discriminants)



Exponentiation Results (96-bit Discriminants)



Exponentiation Results (112-bit Discriminants)



SuperSPAR

SuperSPAR is an integer factoring algorithm based on arithmetic in the ideal class group of imaginary quadratic integers.

- Extends SPAR using a bounded primorial steps search.
- Uses improvements to ideal class group arithmetic and 2,3 representations of power primorials.
- Fastest median time for integers 49-bits to 68-bits of size.
- Fastest average time for integers 49-bits to 64-bits of size.

Median times do not appear in thesis.

Theoretically Optimal Parameters for Exponentiation Stage

$\log_2 N$	$r = \sqrt{\ln N / \ln \ln N}$	$\lfloor N^{1/2r} \rfloor$	p_t	t
16	2.14693	13	13	6
32	2.67523	63	61	18
48	3.08112	221	211	47
64	3.42016	655	653	119
80	3.71609	1738	1733	270
96	3.98139	4258	4253	583
112	4.22355	9802	9791	1208
128	4.44745	21473	21467	2407

Theoretically Optimal Search Bounds

$\log_2 N$	$\lfloor N^{1/2r} \rfloor$	$2m\phi(P_w)$	$m^2\phi(P_w)P_w$	m	w
16	13	16	480	1	3
32	63	64	7680	4	3
48	221	224	94080	14	3
64	655	656	806880	41	3
80	1738	1824	7277760	19	4
96	4258	4320	40824000	45	4
112	9802	10080	222264000	105	4
128	21473	22080	1173110400	23	5

Factorization of the Order

Example #1

$N = 92233754336196605 27, k = 1, \Delta = -kN$

- $\mathrm{ord}([\mathfrak{p}_{19}]) = 2^2 \cdot 13 \cdot 2770667$
- $\mathrm{ord}([\mathfrak{p}_{37}]) = 2^3 \cdot 3 \cdot 13 \cdot 2770667$
- $\mathrm{ord}([\mathfrak{p}_{43}]) = 2^4 \cdot 3 \cdot 13 \cdot 2770667$
- $\mathrm{ord}([\mathfrak{p}_{47}]) = 2^3 \cdot 3 \cdot 13 \cdot 2770667$
- $\mathrm{ord}([\mathfrak{p}_{59}]) = 2^4 \cdot 3 \cdot 13 \cdot 2770667$

where $\mathfrak{p}_p = [p, (b + \sqrt{\Delta})/2]$.

Each prime ideal split N.

Factorization of the Order

Example #2

$N = 182782835644284671 83, k = 3, \Delta = -4kN$

- $\mathrm{ord}([\mathfrak{p}_{11}]) = 2 \cdot 3 \cdot 59 \cdot 157 \cdot 1451$
- $\mathrm{ord}([\mathfrak{p}_{17}]) = 2 \cdot 5^2 \cdot 59 \cdot 157 \cdot 1451$
- $\mathrm{ord}([\mathfrak{p}_{23}]) = 2 \cdot 3 \cdot 5^2 \cdot 59 \cdot 157 \cdot 1451$
- $\mathrm{ord}([\mathfrak{p}_{29}]) = 2 \cdot 3 \cdot 59 \cdot 157 \cdot 1451$
- $\mathrm{ord}([\mathfrak{p}_{31}]) = 2 \cdot 5 \cdot 59 \cdot 157 \cdot 1451$

where $\mathfrak{p}_p = [p, (b + \sqrt{\Delta})/2]$.

Only $[\mathfrak{p}_{23}]$ and $[\mathfrak{p}_{29}]$ split N.

Reusing the Order

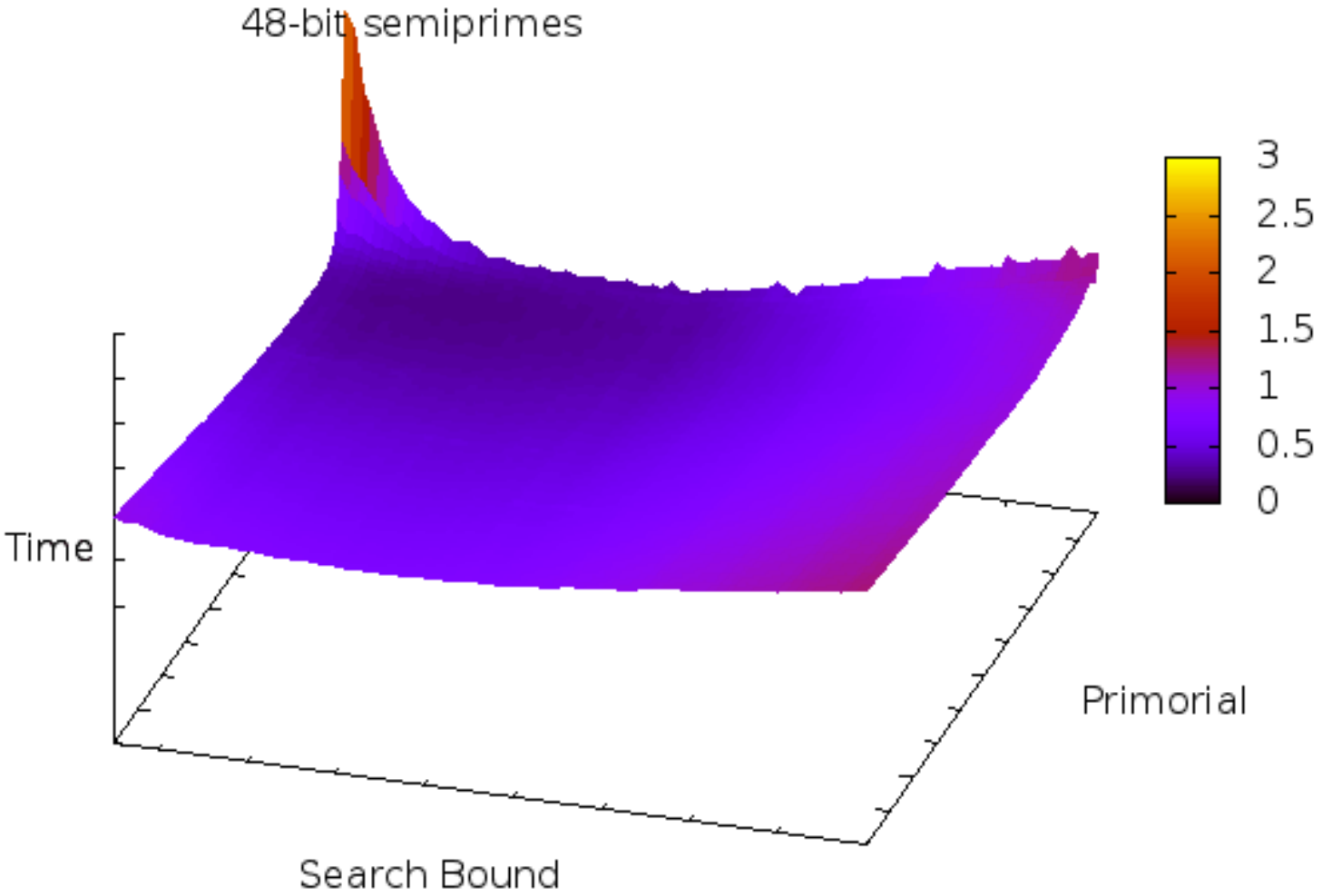
Once the order of an ideal class is known, we skip the search phase.

- The exponentiation stage removed all small primes $\Rightarrow$ search stage not necessary.
- The exponentiation stage did not remove all small primes $\Rightarrow$ stepping coprime will never find the order.

For $N \leq 2^{80}$, this is expected to work better than 97.7% of the time.

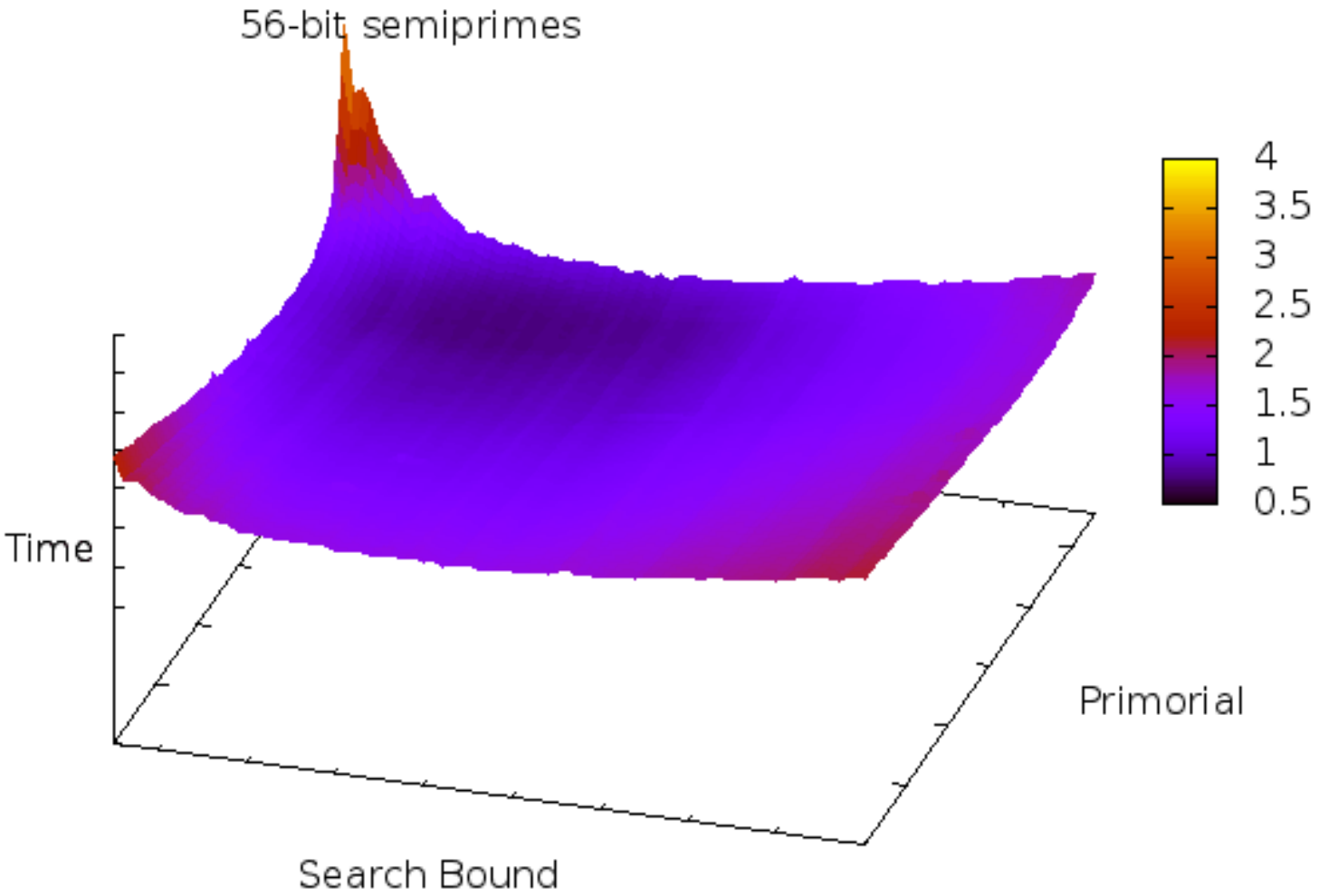
SuperSPAR Search Space

48-bit semiprimes



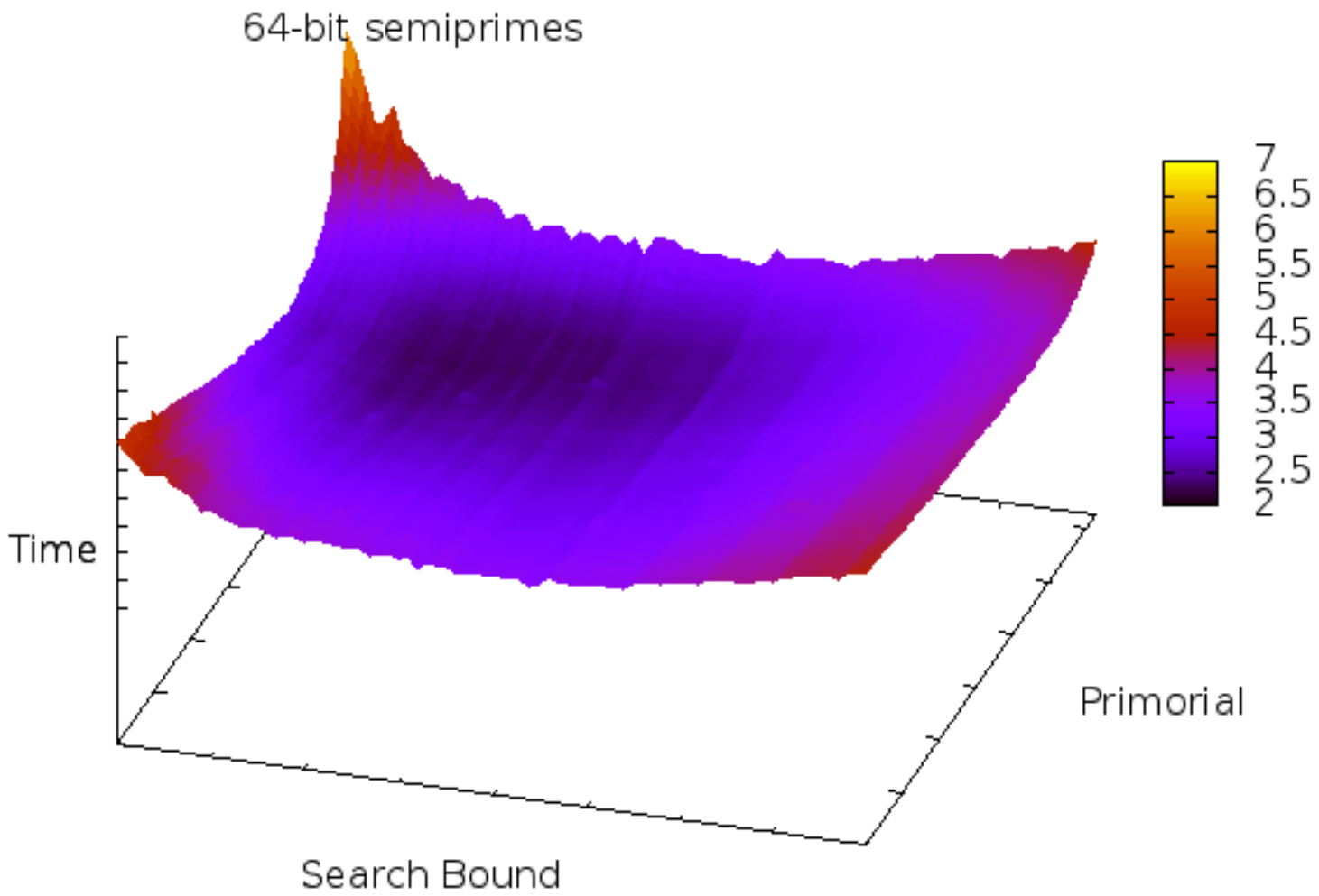
SuperSPAR Search Space

56-bit semiprimes



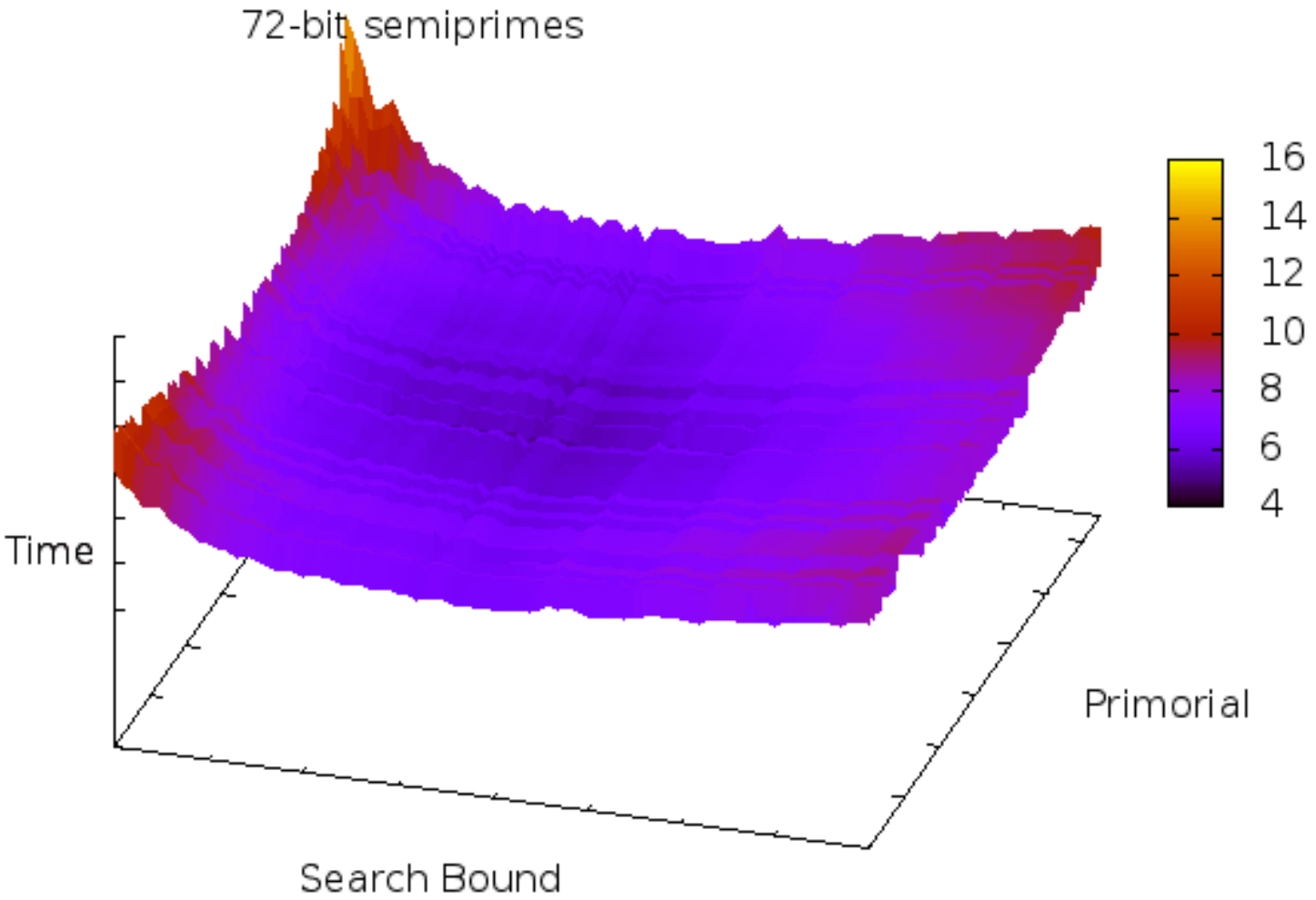
SuperSPAR Search Space

64-bit semiprimes



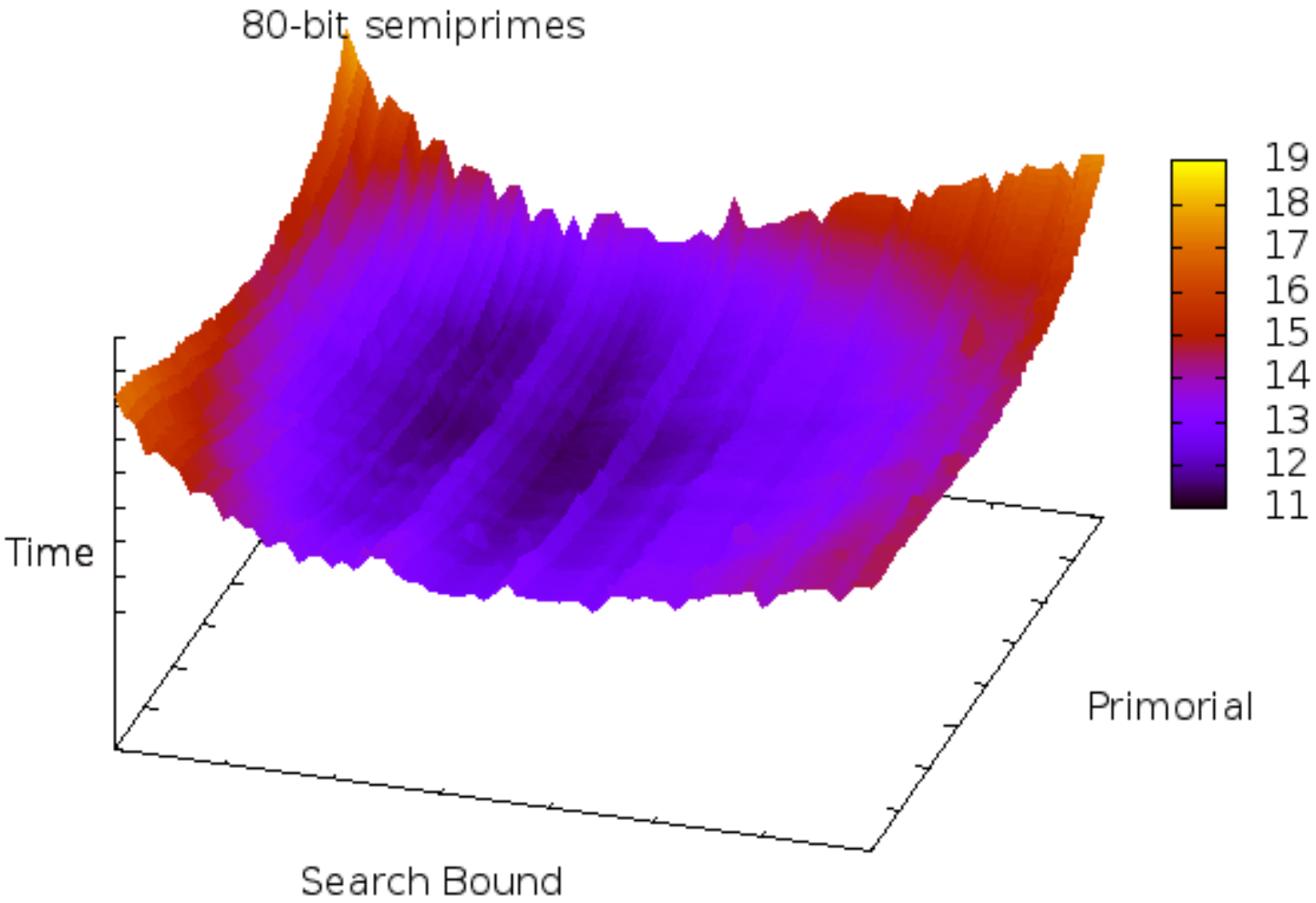
SuperSPAR Search Space

72-bit semiprimes



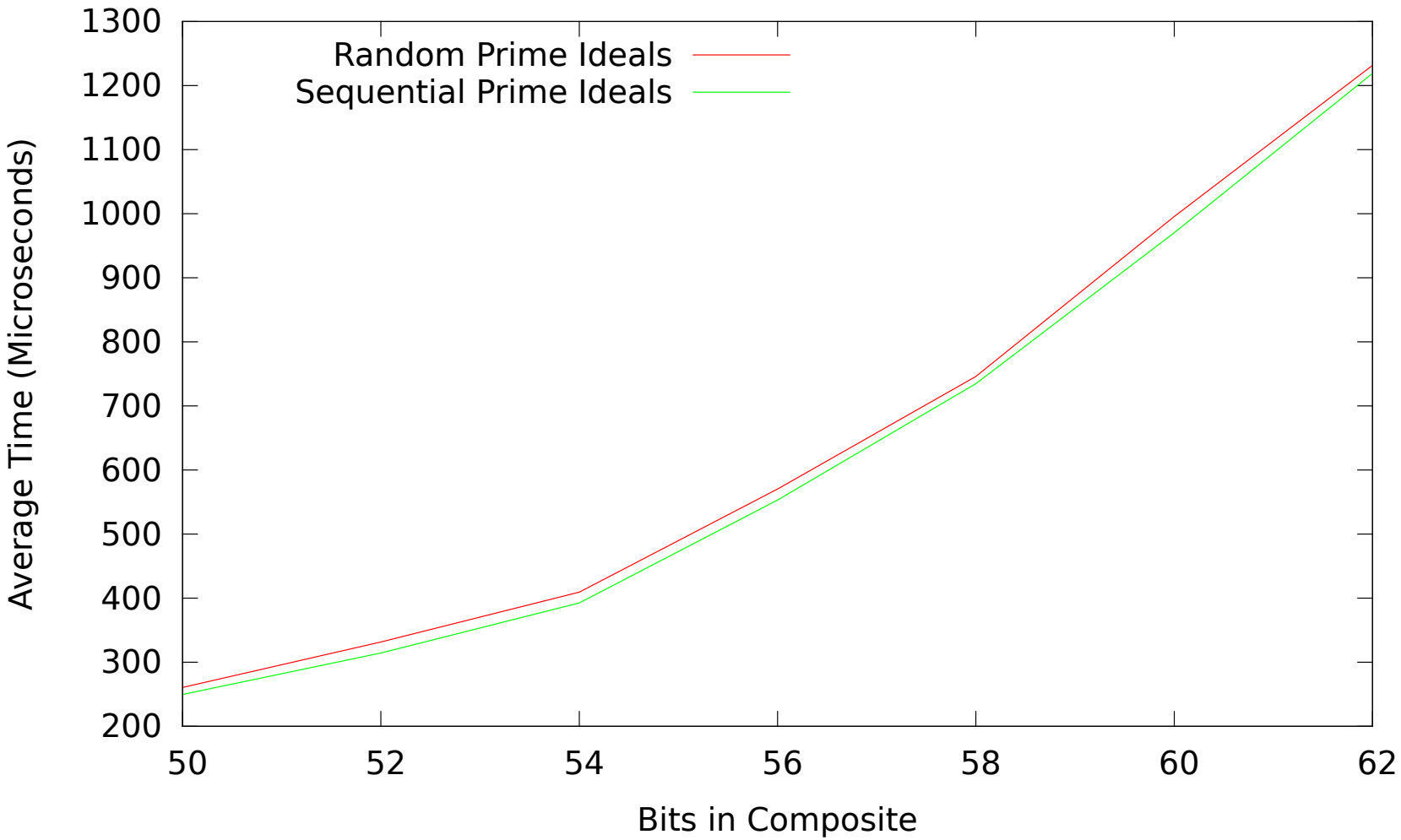
SuperSPAR Search Space

80-bit semiprimes



SuperSPAR Prime Ideals

Sequential vs Random

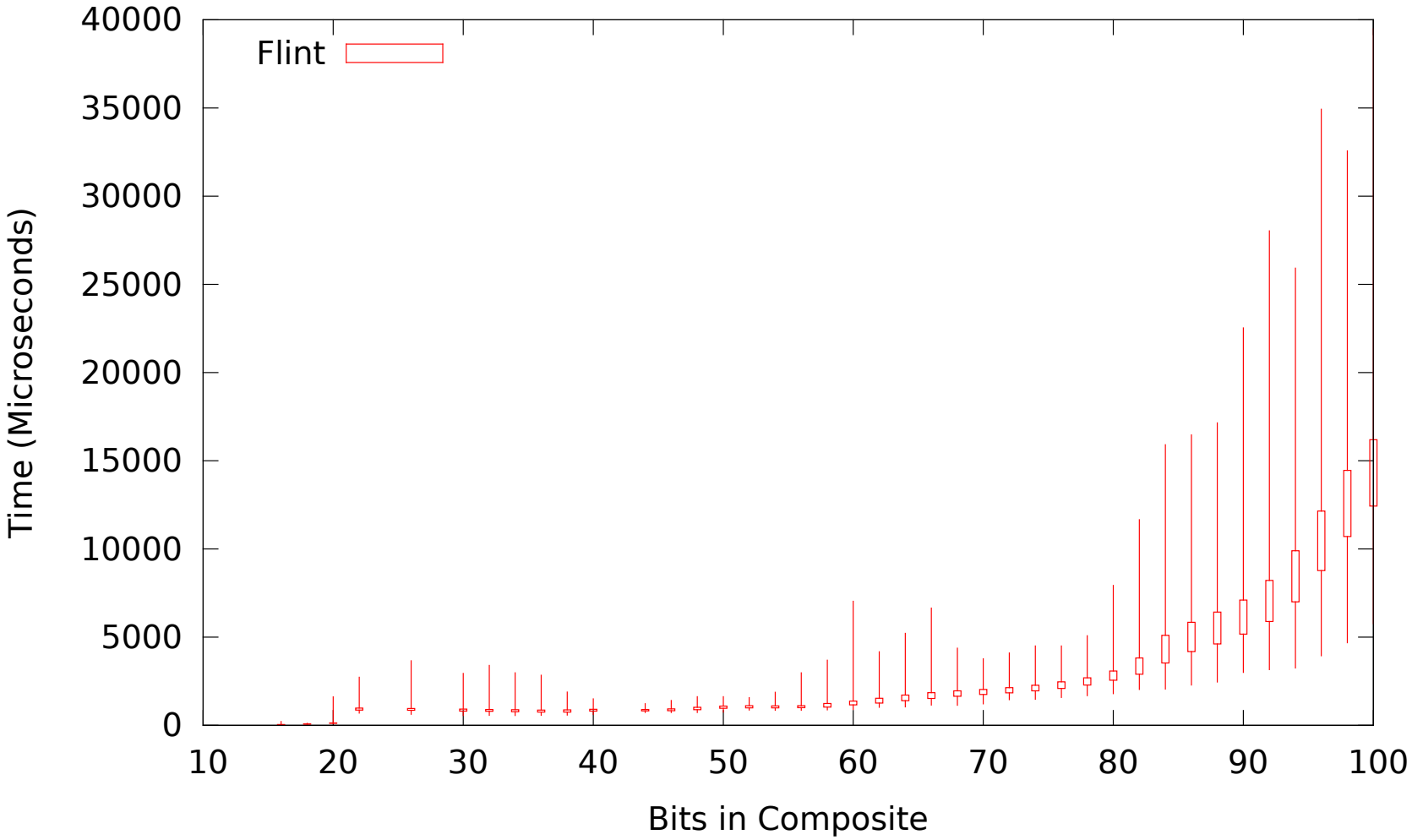


SPAR – Expanding the 2-Sylow Group

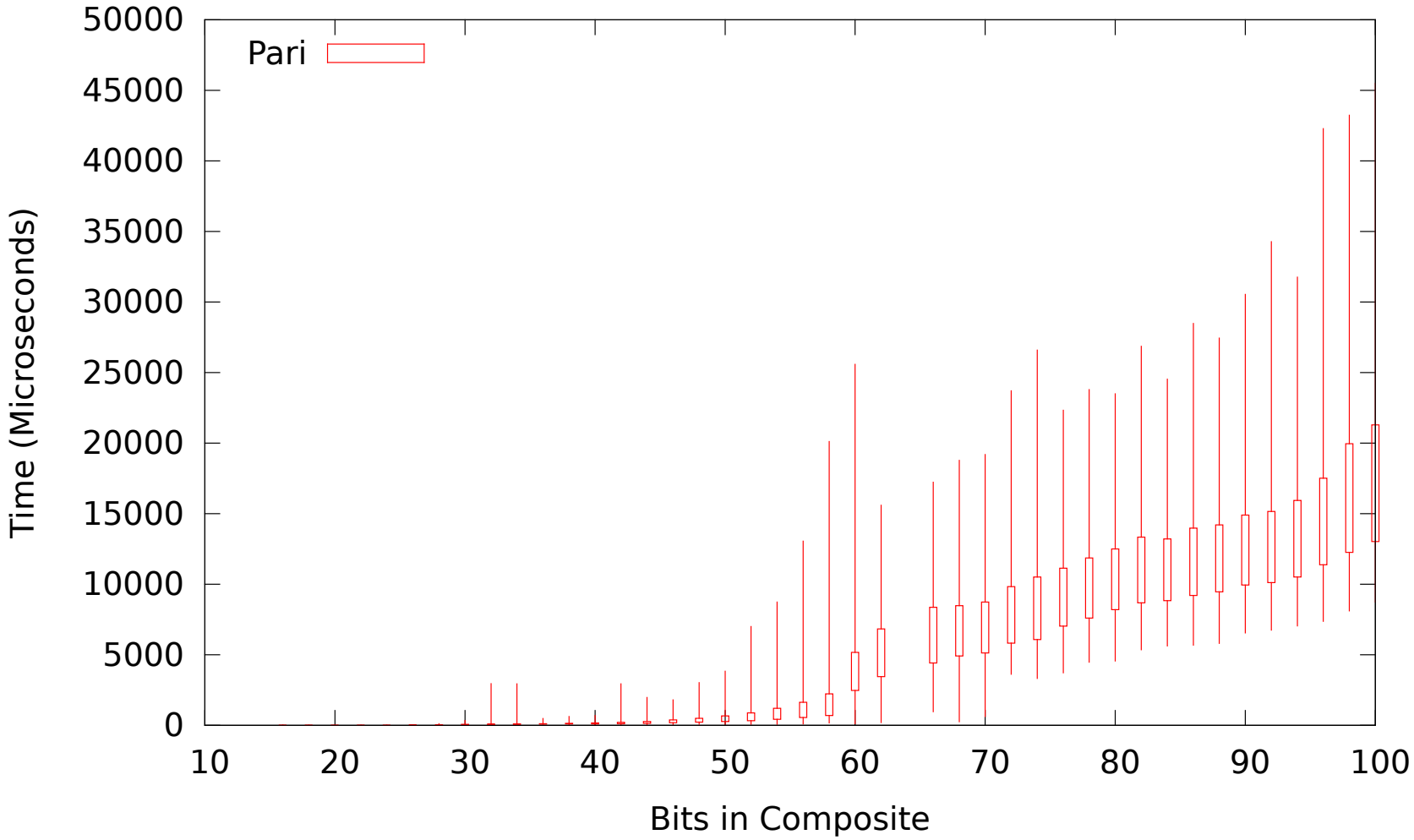
Bits	With 2-Sylow Group	Without 2-Sylow Group
16	48.05208	48.02160
20	84.20318	84.09526
24	197.02479	196.84587
28	463.70674	463.60538
32	937.57875	935.81785
36	1709.55629	1706.32960
40	3255.31447	3248.67998

Table: Average time in microseconds to factor using SPAR when the number of ideals per class group is bound by the size of the integer to factor.

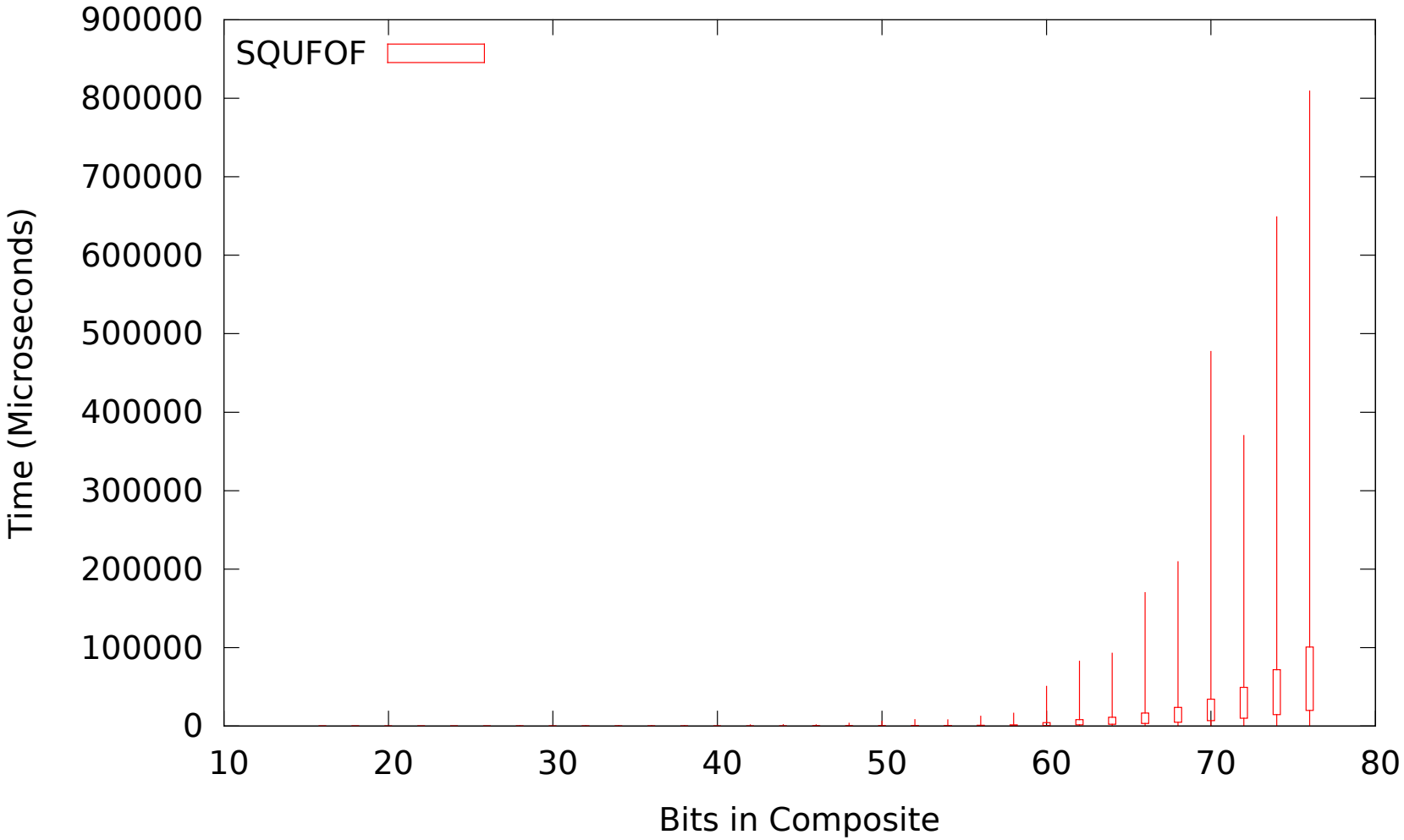
Flint


Flint
Time (Microseconds)
Bits in Composite
0
5000
10000
15000
20000
25000
30000
35000
40000
10
20
30
40
50
60
70
80
90
100

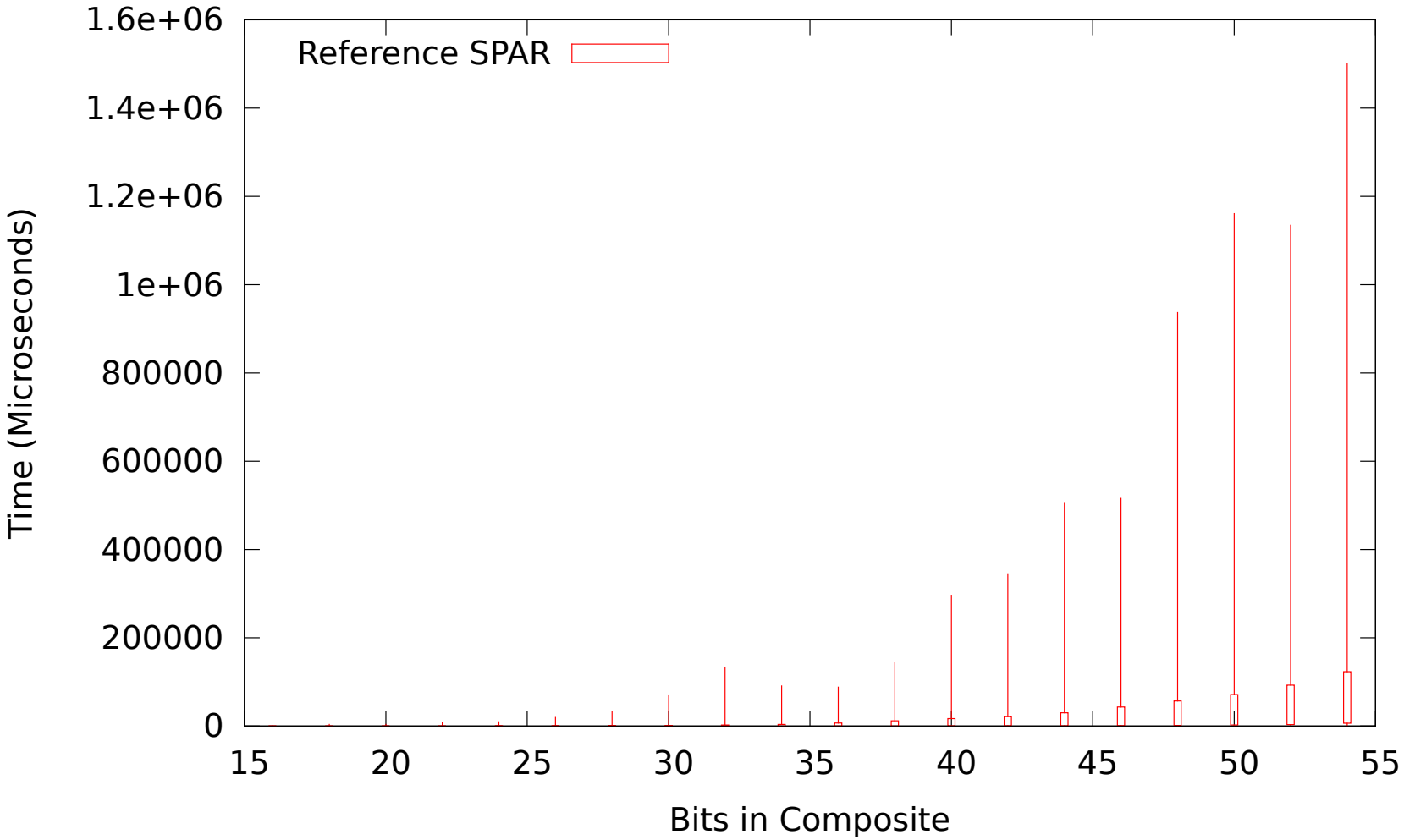
Pari


Pari
Time (Microseconds)
Bits in Composite
50000
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
10
20
30
40
50
60
70
80
90
100

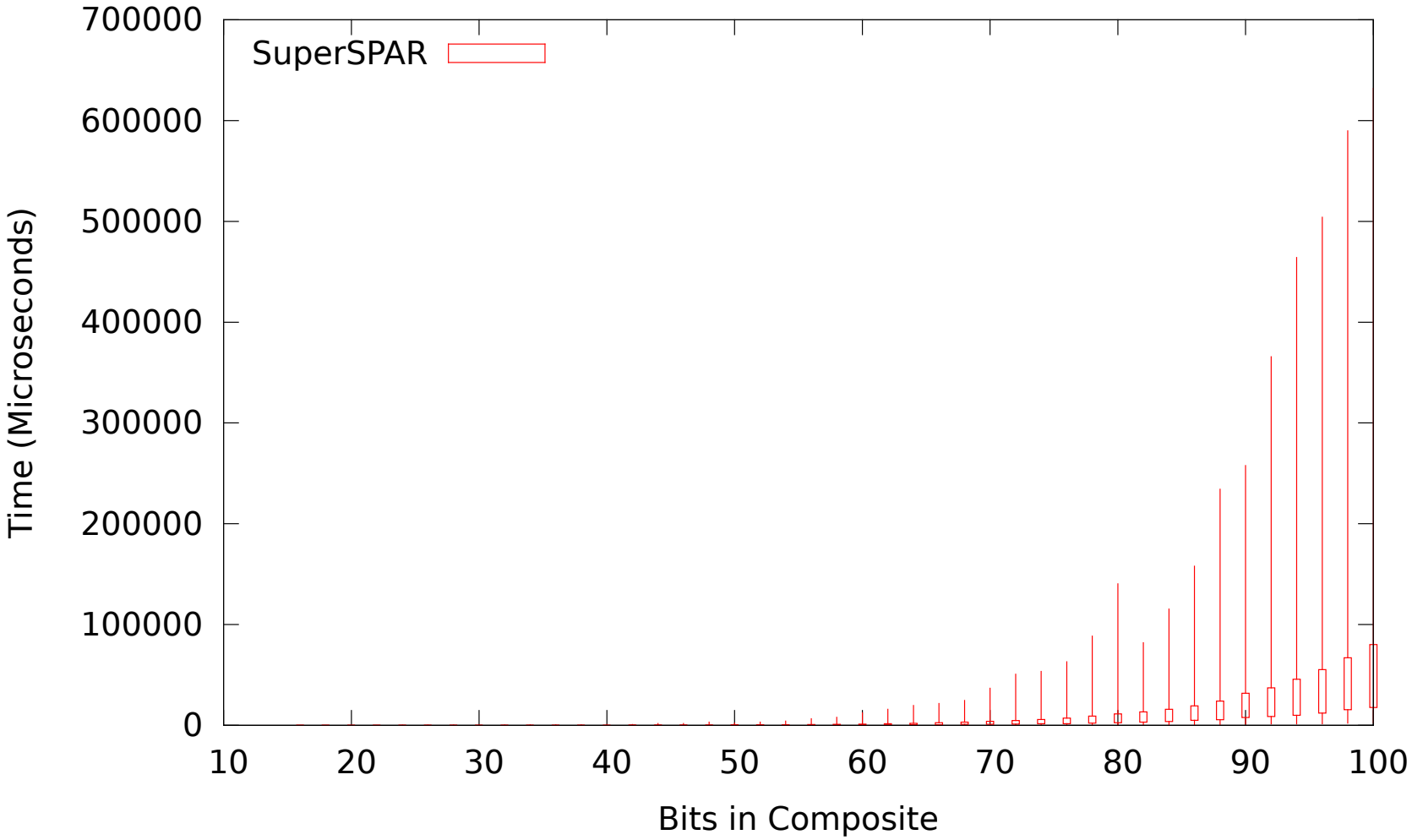
SQUFOF (Optimized from Pari)



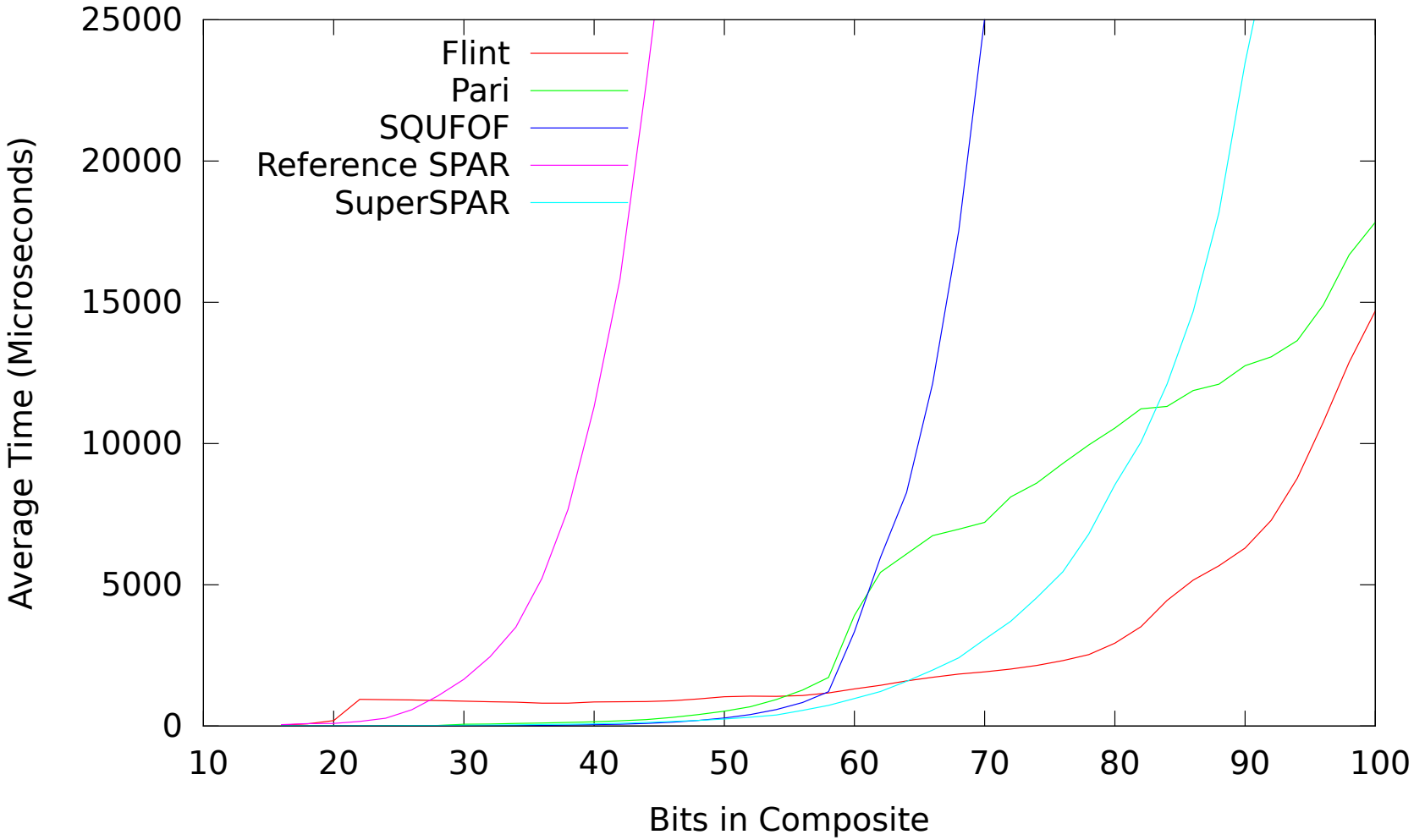
Vanilla SPAR



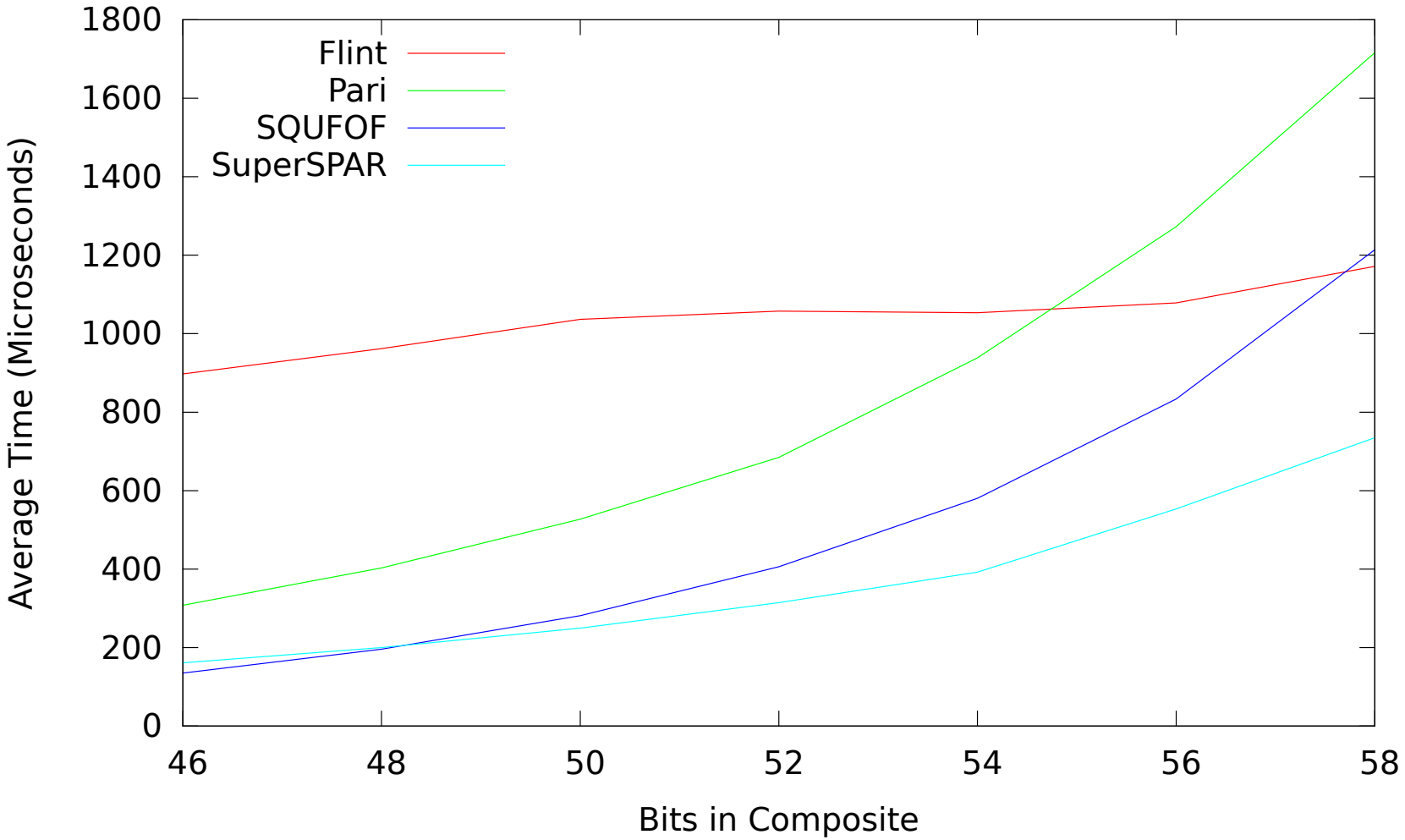
SuperSPAR


SuperSPAR
Time (Microseconds)
700000
600000
500000
400000
300000
200000
100000
0
10
20
30
40
50
60
70
80
90
100
Bits in Composite

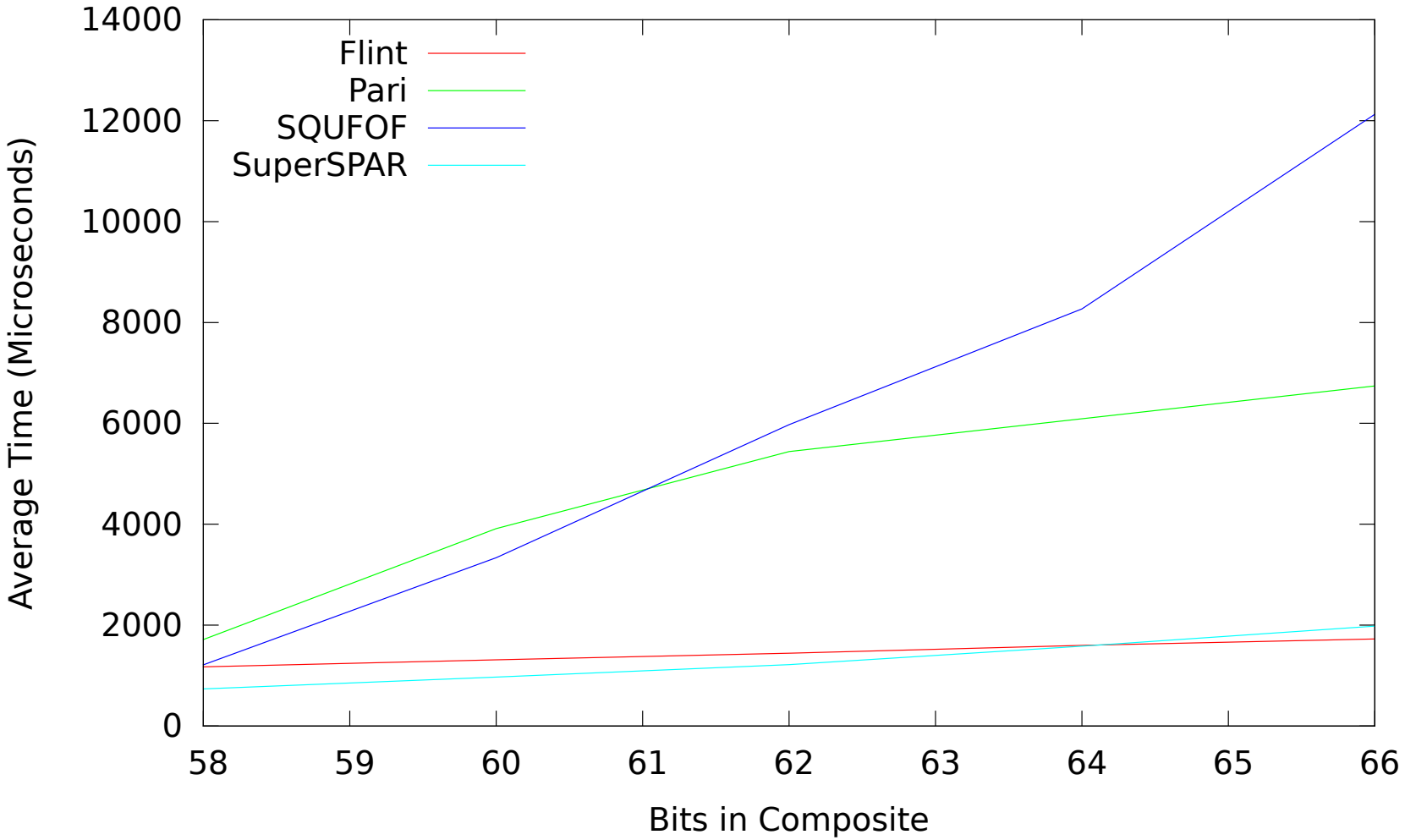
Average Integer Factoring Times



Average Integer Factoring Times (Zoomed Left)



Average Integer Factoring Times (Zoomed Right)



Simple Continued Fraction Expansion of K/L

Recurrences:

$$\begin{aligned} R_i &= R_{i-2} - q_i R_{i-1} \\ C_i &= C_{i-2} - q_i C_{i-1} \\ A_i &= A_{i-2} + q_i A_{i-1} \\ B_i &= B_{i-2} + q_i B_{i-1} \end{aligned}$$

Invariants:

$$\begin{aligned} C_i &= (-1)^{i+1} B_i \\ L &= R_i B_{i-1} + B_i R_{i-1} \\ (-1)^{i-1} &= A_i B_{i-1} - B_i A_{i-1} \\ (-1)^{i+1} R_i &= L A_i - K B_i \end{aligned}$$