Some Remarks on

Polynomial Selection in the GNFS

Colin Stahlke, Christine Priplata

{stahlke,priplata} at edizone.de

1. Basics
2. Improving local optimization
3. Measuring the quality
4. Improving a quality function
5. Summary

1. Basics
2. Improving local optimization
3. Measuring the quality
4. Improving a quality function
5. Summary

Basics: The polynomial pair in the GNFS

Choose two irreducible, coprime polynomials $f_{1,} f_2 \in \mathbb{Z}[x]$ such that

$$\exists \quad m \in \mathbb{Z}: \quad f_1(m) \equiv f_2(m) \equiv 0 \qquad mod\ N$$

Let F_1 and F_2 be the homogenized polynomials of f_1 and f_2.

Now we sieve for pairs $(a, b) \in \mathbb{Z}^2$ such that F_i(a,b) has a smooth prime ideal decomposition in the number field $\mathbb{Q}[x]/(f_i(x))$ for i=1,2.

A good polynomial selection is crucial for the size and the speed of the GNFS.

Basics: Constructing polynomial pairs

It is crucial to get small coefficients.

Expansion to base m:

$$N = \sum_{i=0}^{d} a_i m^i \qquad f_1(x) = \sum_{i=0}^{d} a_i x^i \qquad f_2(x) = x - m$$

If the degree d is larger, the coefficients a_i are smaller.

- There are methods to control the size of some of the a_i.
- Extensive search reduces the size of all a_i.
- The leading coefficient of f_2 can be larger than 1.

Finally local optimization.

Basics: Measuring the quality of polynomials

The best test for the quality of (f_1,f_2) is sieving.

$Q(f_1, f_2) = \#\{ (a, b) \in \mathbb{Z} \mid (a, b) = 1,\ F_i(a, b) \text{ is } L_i\text{-smooth}, |a| \leq A, 0 < b \leq B \}$

This test is slow. For faster approximations we need the following notations:

prime p *small* <==> p<1000

K(n) := product of small primes (with multiplicity) dividing n

$\alpha(F) := E_{n\in\mathbb{Z}}(\log(K(n))) - E_{coprime\,(a,b)\in\mathbb{Z}^2}(\log(K(F(a,b))))$ E expectation value

$\rho(x)$:= Dickmann ρ-Function (probability that a number of size n is $n^{1/x}$-smooth)

Then the probability that F(a,b) is L-smooth is about:

$$\rho\left(\frac{\alpha(F)+\log(F(a,b))}{\log(L)}\right)$$

Basics: Measuring the quality of polynomials

$$Q_1(f_1, f_2) = \frac{6}{\pi^2} \int_{\substack{|a| \le A \\ 0 < b \le B}} \varrho\left(\frac{\alpha(F_1) + \log F_1(a,b)}{\log L_1}\right) \varrho\left(\frac{\alpha(F_2) + \log F_2(a,b)}{\log L_2}\right) \, da \, db$$

$$Q_2(f_1, f_2) =$$

$$\int_0^\pi \varrho\left(\frac{\alpha(F_1) + \log F_1(-A\cos\theta, B\sin\theta)}{\log L_1}\right) \varrho\left(\frac{\alpha(F_2) + \log F_2(-A\cos\theta, B\sin\theta)}{\log L_2}\right) \, d\theta$$

$$Q_3(f_1) = \alpha(F_1) + \frac{1}{2} \log \left(\int_{\substack{|a| \le A \\ 0 < b \le B}} F_1(a,b)^2 \, da \, db \right)$$

$$Q_4(f_1) = \max_{0 \le i \le d_1} |a_i| s^{i - \frac{d_1}{2}}$$

s = A/B skewness

Basics: Local optimization

$$Q_3(f_1) = \alpha(F_1) + \frac{1}{2}\log\left(\int_{\substack{|a|\le A\\ 0<b\le B}} F_1(a,b)^2\,da\,db\right)$$

local part $\alpha(F_1)$ + infinite part (think of $Q_4(f_1) = \max_{0\le i\le d_1} |a_i| s^{i-\frac{d_1}{2}}$)

$$\alpha(F_1) = \sum_{p \text{ small prime}} \alpha_p(F_1)$$

Replace f_1 by $f_1+(ax+b)f_2$ to optimize the local part $\alpha(F_1)$.

Translate f_1 and f_2 to reduce the infinite part.

Choosing (a,b) in some congruence classes modulo small primes p makes

$\alpha_p(F_1)$ small and speeds up the process.

1. Basics
2. Improving local optimization
3. Measuring the quality
4. Improving a quality function
5. Summary

Improving local optimization: 768 bit

N = 1230186684530117755130494958384962720772853569595334792197322
4521517264005072636575187452021997864693899564749427740638459
2519255732630345373154826850791702612214291346167042921431160
2221240479274737794080665351419597459856902143413

After 30 CPU years of T. Kleinjung's first polynomial selection:

$\alpha = -7{,}30$

$$Q_2 = 3{,}79 \cdot 10^{-9}$$

skewness = 44204,72

Now we tried with T.Kleinjung's second polynomial selection

Task: locally optimize 3403 polynomial pairs got after 1 CPU day.

After 26 CPU days of ordinary local optimization (without congruence classes):

$\alpha = -7{,}20$

$$Q_2 = 2{,}18 \cdot 10^{-9}$$

Improving local optimization: 768 bit

First a quick and dirty test with lots of congruences, skipping loads of (a,b) pairs.

After 11 CPU minutes of local optimization with congruences for 2,3,5,...,19:

$\alpha = -7{,}70$

$$Q_2 = 2{,}35 \cdot 10^{-9}$$

skewness = 2124936

A more thorough approach with more reasonable parameters gave after 9 CPU hours:

$\alpha = -8{,}329$

$$Q_2 = 2{,}57 \cdot 10^{-9}$$

Improving local optimization: 768 bit

Task: locally optimize 16.912.909 polynomial pairs got after several CPU years within some CPU days.

First idea: sort polynomial pairs by their infinite part and consider only the best.

failed

Improving local optimization: 768 bit

Task: locally optimize 16.912.909 polynomial pairs got after several CPU years within some CPU days.

First idea: sort polynomial pairs by their infinite part and consider only the best.

failed

Second idea: We must be quick and dirty again!

After 393 CPU hours of local optimization with congruences for 2,3,5,...,23:

$\alpha = -8{,}783$

$$Q_2 = 3{,}53 \cdot 10^{-9}$$

But: How to do the thorough approach now?

Improving local optimization: 768 bit

T. Kleinjung did several CPU years of local optimization
with congruences for 2,3,5,...,13. Best result:

$\alpha = -8{,}99$

$$Q_2 = 3{,}81 \cdot 10^{-9}$$

We have access to the 1059 best pairs. They are all local optimizations
from just 2 polynomial pairs.

Already detected by our quick and dirty search!
Therefore: local optimization of e.g. only 5 polynomial pairs.

After some CPU hours of local optimization of only the best pair, we got:

$$Q_2 = 3{,}79 \cdot 10^{-9}$$

Improving local optimization: 768 bit

Result

For T.Kleinjung's second polynomial selection:

1. Do a quick search with many congruences.
2. Search through the best results much more carefully.

old method	same quality	new method
several CPU years of local optimization	⟷	400 CPU hours of local optimization

1. Basics
2. Improving local optimization
3. Measuring the quality
4. Improving a quality function
5. Summary

Measuring the quality: 350 bit

Probesieben

Experiment 1: Sieve the first 100 special q and note

t_1 the number of relations per second
q_1 the number of relations per special q

Experiment 2: Sieve about every 2000th special q (with different parameters) and note

t_2 the number of relations per second
q_2 the number of relations per special q

We did this for 83824 polynomial pairs,
all of them local optimizations of the same pair.

Are there any correlations between these numbers?

Measuring the quality: 350 bit

Measuring the quality: 350 bit

Measuring the quality: 350 bit

ϱ	t_1	q_1	t_2	q_2
t_1	1	0.9584	0.6768	0.7028
q_1	0.9584	1	0.7373	0.7745
t_2	0.6768	0.7373	1	0.9678
q_2	0.7028	0.7745	0.9678	1

If the numbers were uncorrelated to the total sieving quality, there was no reason why the numbers of different experiments would correlate at all.

Probesieben seems to reflect the quality of polynomial pairs.

Improving a quality function

Linear combination with best correlation

$$Q_3(f_1) = \alpha(F_1) + \frac{1}{2} \log \left(\int_{\substack{|a| \leq A \\ 0 < b \leq B}} F_1(a,b)^2 \, da \, db \right)$$

Is the factor ½ correct?
We want to choose the factor such that Q_3 correlates best with the sieving quality.

Think of k=83824 and n=2.

Let y be a k-tuple and $x_1, x_2, \ldots, x_n$ be n k-tuples.

We want to find the linear combination $\sum_{j=1}^{n} \lambda_j x_j$ that correlates best with y. This can be done by linear algebra.

Let y be the 83824-tuple of values t_2 , x_1 the 83824-tuple of the local parts and x_2 the 83824-tuple of the infinite part of Q_3. Set $\lambda_1=1$, then $\lambda_2=2{,}07$.

Improving a quality function

Linear combination with best correlation

	Exp. 1 #Rel/s	Exp. 1 #Rel/q	Exp. 2 #Rel/s	Exp. 2 #Rel/q	Q_2	Q_3
λ_2	$2,33$	$2,03$	$2,07$	$2,15$	$3,99$	$1,00$

We suggest λ_2=2 and therefore the following new quality function:

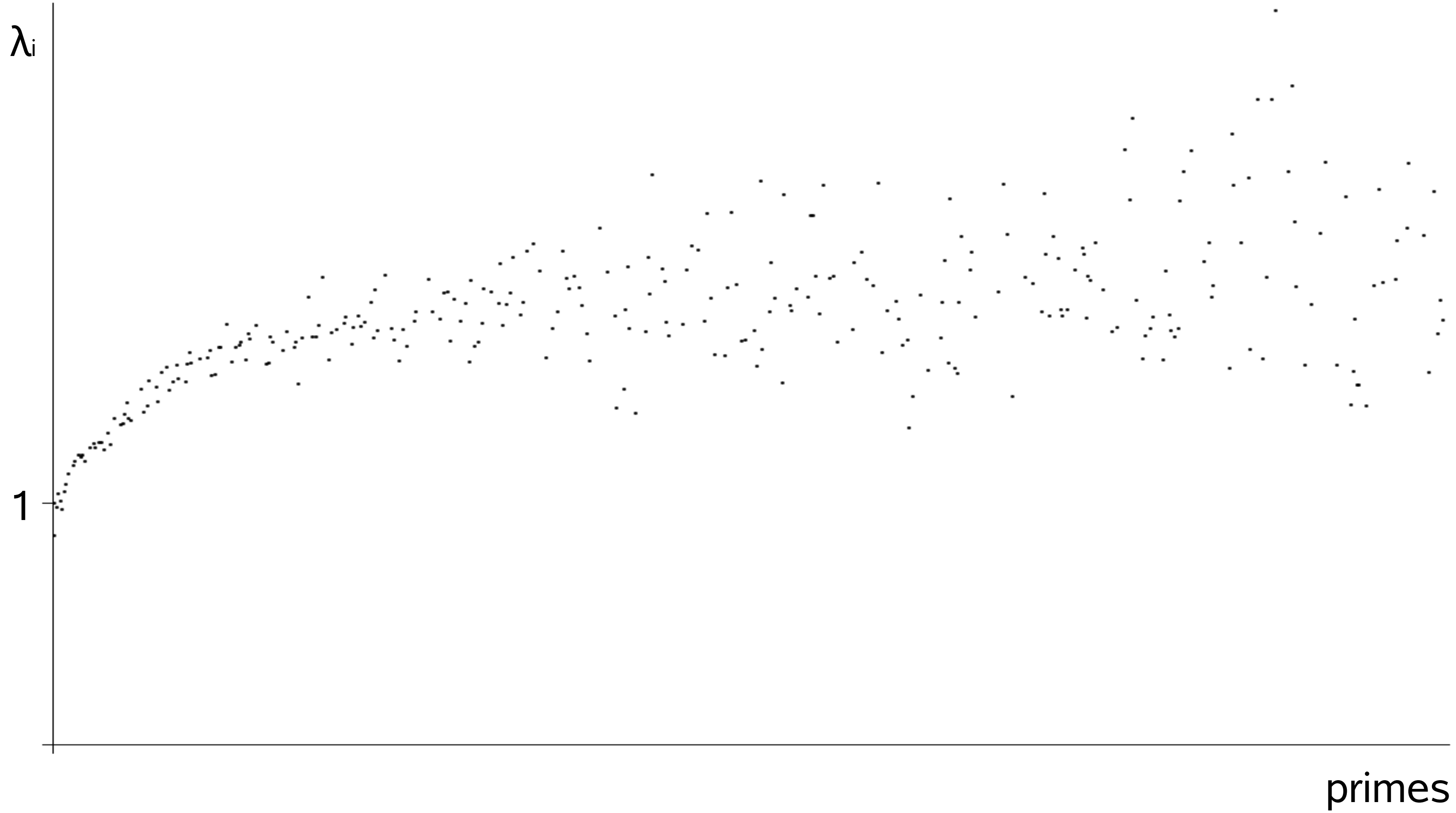
$$Q_3'(f_1) = \alpha(F_1) + \log\left(\int_{\substack{|a|\leq A\\ 0<b\leq B}} F_1(a,b)^2\, da\, db\right)$$

Now we also have: $\alpha(F_1) = \sum_{p\, small\ prime} \alpha_p(F_1)$

Should the summands for p=2,3,5,...,1999 get new weights?
We want to change the local part.

Improving a quality function

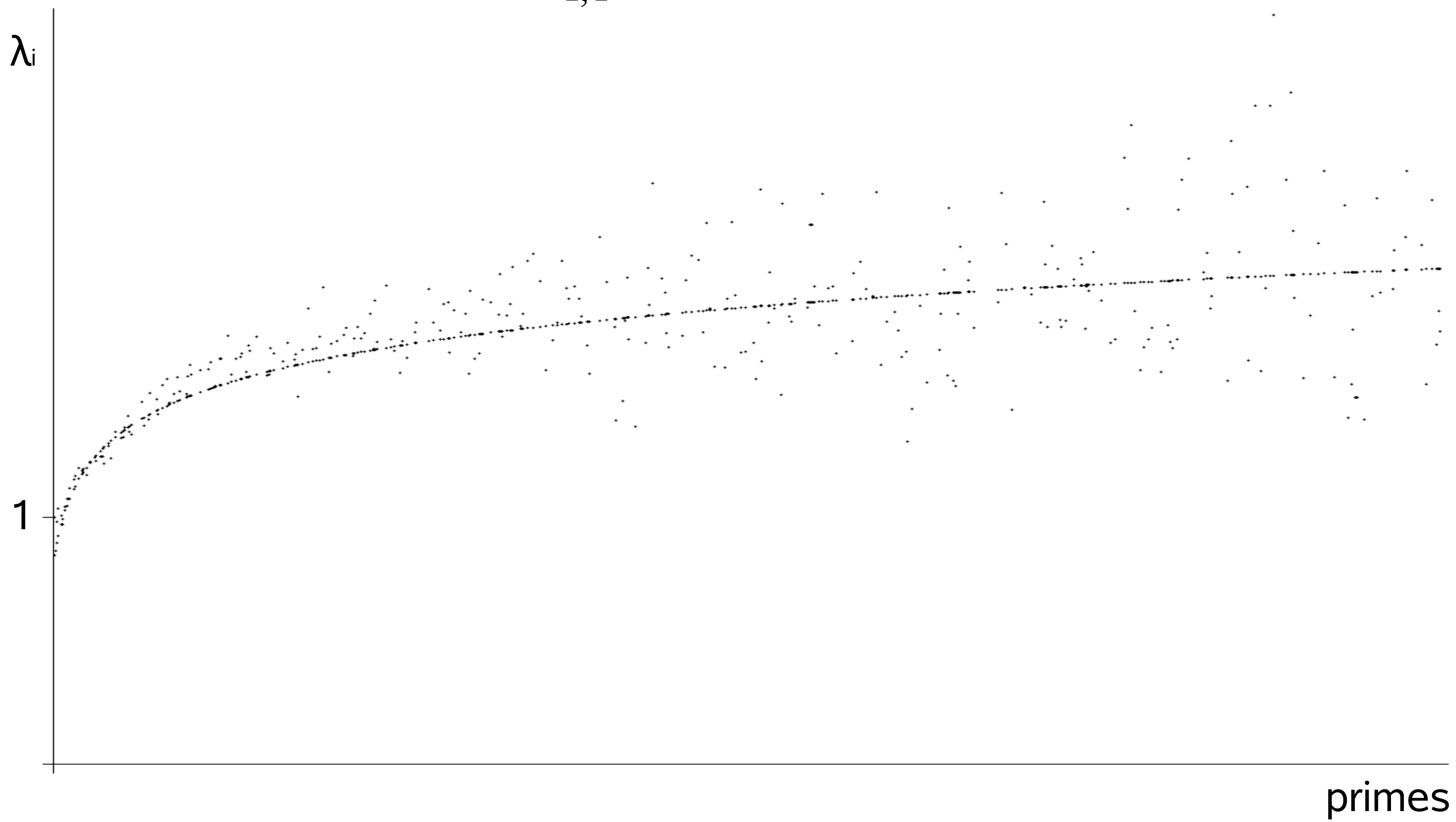
Let y be the 83824-tuple of values t_2 and let the x_j be the 83824-tuples of the local parts $\alpha_p(F_1)$.
Set λ_1=1. We represent the numbers λ_i in the following picture:



Improving a quality function

We approximate the cloud of points by the following function:

$$f(x) = 0.281936 \cdot \log \frac{(x+10)}{4{,}4} \quad \text{for } x>2 \quad \text{and} \quad f(2)=1$$



Improving a quality function

We suggest the following new quality function:

$$Q_3''(f_1) = \frac{1}{2}\log\left(\int_{\substack{|a|\le A\\ 0<b\le B}} F_1(a,b)^2\,da\,db\right) + \sum_p f(p)\cdot\alpha_p(F_1)$$

with $f(x)=0.281936\cdot\log\frac{(x+10)}{4{,}4}$ for x>2 and f(2)=1.

Now we want to test the two new quality functions Q'_3 and Q''_3. We

- choose an arbitrary 350 bit number which is a product of two large primes,
- generate 67774 good polynomial pairs having several different common zeroes,
- perform experiment 1 and 2 with these 67774 polynomial pairs
- and calculate Q_2, Q_3, Q'_3, Q''_3 and the correlations with t_1, q_1, t_2 and q_2.

Which quality functions correlate best with the quality got from real sieving?

Improving a quality function

ϱ	t_1	q_1	t_2	q_2
Q_3	-0.61525	-0.70406	-0.67121	-0.70073
Q_3'	-0.61600	-0.69990	-0.67732	-0.69648
Q_3''	-0.67113	-0.76005	-0.71864	-0.74903
Q_2	0.75045	0.77738	0.74335	0.78090

- Q'_3 is as good as Q_3.
- Q''_3 is much better than Q_3 and takes the same CPU time.
- Q_2 is still more accurate but takes more CPU time.

Conclusion: The quality function Q_3 should be modified in a way similar to Q''_3.

1. Basics
2. Improving local optimization
3. Measuring the quality
4. Improving a quality function
5. Summary

Summary

We presented

- a strategy for speeding up the local optimization part of the polynomial selection process
  - using congruences make T. Kleinjung's most recent polynomial selection work for large bit lengths
  - detecting good polynomial pairs speeds up local optimization considerably, so CPU time can be used to find better polynomials
- an improvement of the quality function Q_3
  - method for choosing linear combinations with best correlation
  - connection between Probesieben and real sieving quality
  - Q''_3

$$Q''_3(f_1) = \frac{1}{2}\log\left(\int_{\substack{|a|\leq A \\ 0<b\leq B}} F_1(a,b)^2\,da\,db\right) + \sum_p f(p)\cdot\alpha_p(F_1)$$