



INTEGRAL MINIMISATION IMPROVEMENT FOR MURPHY'S POLYNOMIAL SELECTION ALGORITHM

Přemysl Jedlička

Abstract

We consider Murphy's polynomial selection algorithm for the general number field sieve. One of the steps in this algorithm consists of finding a minimum of an integral. However, the size of the polynomial coefficients causes the classical steepest descent algorithm to be ineffective. This article brings an idea how to improve the steepest descent algorithm so that it converges better and faster.

Most of today's security applications are based on our inability to quickly factorise integers. The fastest known algorithm for splitting a large integer into the product of primes is the Number Field Sieve [3], [6]. In this algorithm, we work in a number field $K = \mathbb{Q}[x]/f$, where $f \in \mathbb{Z}[x]$ is an irreducible polynomial that has a root modulo the factorised number.

It is a difficult task to find such a polynomial f . More precisely said, there are infinitely many such polynomials but it is difficult to find good ones; one of the features of a good polynomial is to have small coefficients. Not much was known about looking for such good polynomials since Brian Murphy presented a method for polynomial selection in his thesis [5]. Although there now exist newer methods [2], this one is still widely used. A way how to achieve small coefficients of the sought polynomial is to define a multivariate function based on a definite integral over the polynomial and to find the minimum. Unfortunately, there was given no hint how to find it.

Key Words: Number field sieve, Polynomial selection, Skewed polynomials, Steepest descent

2010 Mathematics Subject Classification: 12H04, 65H04

Received: September, 2009

Accepted: January, 2010

The most natural approach is to use the steepest descent method and it is actually used in many implementations. However, the polynomial f has very large coefficients (typically about 25 digits) and this brings many difficulties: the method does not need to converge; and when it does, it can do it so very slowly.

This article describes how to deal with this special problem. We benefit from the special form of the function we are working with – it is in fact a paraboloid with respect to some (the most critical) variables and hence we can compute directly the minimum when the other variables are fixed. If this correction is done between each step of the steepest descent algorithm, it converges faster.

1 Problems appearing in the steepest descent implementations

Let us denote by N the big number we want to factorise using the number field sieve. In Murphy's thesis [5], page 84, there is the following construction: we have a polynomial f_m with a root m modulo N . We perform slight changes to the polynomial, called a translation by a number t and a rotation by a polynomial P :

$$f(x) = f_m(x - t) + P(x)(x - t - m).$$

Then we consider the homogeneous polynomial associated with f

$$F(x, y) = y^d f\left(\frac{x}{y}\right).$$

We would like to “minimise” the values of the polynomial over some area, without loss of generality a rectangle of area 1. That means, we want to find optimal values of P , t and s so that the integral

$$\int_{-\sqrt{s}^{-1}}^{\sqrt{s}^{-1}} \int_{-\sqrt{s}}^{\sqrt{s}} F^2(x, y) dx dy$$

is minimal. This is done by a multi-variate minimisation algorithm.

It is probable that every implementation of this problem should use the steepest descent method somehow. However, the direct implementation of this method is not very good. The coefficients of the polynomial f_m have many digits and hence some of the coefficients of F^2 are extremely large (about 10^{50}). Such sizes are not common in everyday numerical analysis.

Two types of problems can occur. In both, informally said, we have a valley, or more precisely a very deep canyon and we start our minimisation algorithm on a side of this valley (canyon). The side is very steep and therefore the differential shows we should continue right downhill (see Figure 1).

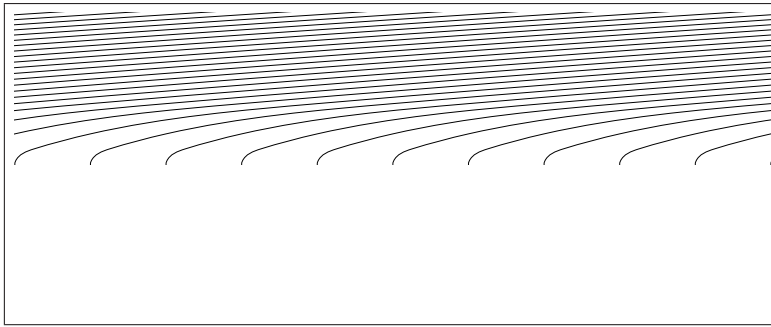


Figure 1: Very deep valley in the multivariate polynomial (using contour lines)

The worst situation is when the valley is “narrower” than the minimal step. Then by making a step (even the minimal one) in the direction of the differential, we jump over the valley, eventually reaching a spot on the opposite side of the valley that is higher than the spot of the origin. Thus the algorithm ends here, far from finding a suitable minimum.

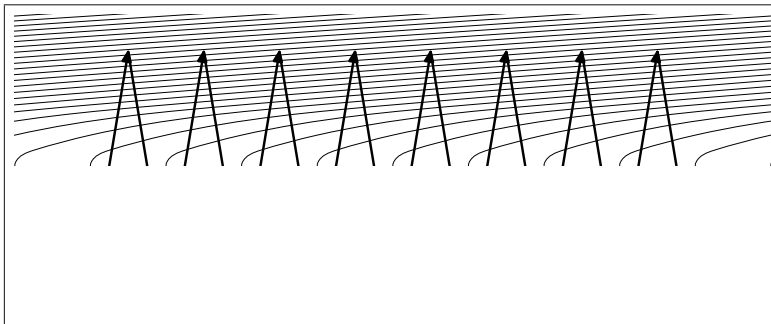


Figure 2: A slow convergence in the valley

The better situation when the valley is not that extremely narrow. We make a step across the valley ending on the other side in a lower spot. Now we have to turn back and cross the valley again ending close to the spot of origin (see Figure 2). Nevertheless, we keep getting lower hence there is no need to stop the algorithm or alter the step length. We continue zig-zagging and we eventually converge to a local minimum but after a very long time. It does not help if we eventually try to change the step length. The sides are much steeper than the valley itself and hence we do zig-zagging always unless

we are really at the bottom of the valley. However, even if we happen to be there in one step, there is no guarantee we will not be on a side of the valley after the next step.

Let us take a look, *e.g.*, into the source code of the GGNFS implementation [4] of the general number field sieve. It happens many times there that the steepest descent routine returns the same values as the ones we started with (due to the first problem). The authors tried to deal with it by “perturbing” the found values, that means by changing the values and trying luck but as one can imagine, it was not always successful.

2 Solution of the problem

Fortunately, the form of the function , we are dealing with, enables us to better solve the problem. Actually, it is the coefficients of the rotation polynomial $P(x)$ that bring all the mess into the steepest descent algorithm. Let us take, $P(x) = c_2x^2 + c_1x + c_0$, for instance, the process will be analogical for different polynomials. We have

$$f(x) = f_m(x - t) + (c_2x^2 + c_1x + c_0)(x - t - m).$$

We denote by S the rectangle $[-\sqrt{s}, \sqrt{s}] \times [-\sqrt{s}^{-1}, \sqrt{s}^{-1}]$ and by F_S the integral

$$\iint_S F^2(x, y) dx dy.$$

It is a function in five variables: c_0, c_1, c_2, s and t . From the construction of the function F_S , we easily deduce the decomposition

$$F_S = B(t, s) + \sum_{i=0}^2 B_i(t, s)c_i + \sum_{i=0}^2 \sum_{j=i}^2 B_{ij}(t, s)c_i c_j$$

where B, B_i and B_{ij} for $0 \leq i \leq j \leq 2$ are functions in variables t and s only.

Let us fix t and s and denote by b_{ij} the value of B_{ij} in these fixed values. The graph of the function forms a quadric in \mathbb{R}^4 , actually it should be a 4-dimensional paraboloid. It is easy to compute the minimum of this paraboloid using partial derivatives:

$$\begin{aligned} \frac{\partial F_S}{\partial c_0} &= b_0 + 2b_{00}c_0 + b_{01}c_1 + b_{02}c_2 \\ \frac{\partial F_S}{\partial c_1} &= b_1 + b_{01}c_0 + 2b_{11}c_1 + b_{12}c_2 \\ \frac{\partial F_S}{\partial c_2} &= b_2 + b_{02}c_0 + b_{12}c_1 + 2b_{22}c_2 \end{aligned}$$

The minimum is the triple (c_0, c_1, c_2) for which all three partial derivatives are equal to 0. Therefore we have to solve the system of linear equations given by the matrix

$$\left(\begin{array}{ccc|c} 2b_{00} & b_{01} & b_{02} & -b_0 \\ b_{01} & 2b_{11} & b_{12} & -b_1 \\ b_{02} & b_{12} & 2b_{22} & -b_2 \end{array} \right)$$

which can be done using Gaussian elimination.

The algorithm of the steepest descent is now the following: at each step, we use the values of t and s to compute the numbers b_i and b_{ij} . Then we compute the values of c_i and we obtain a point where the partial derivatives with respect to c_i should be 0, for all i (and we should be on the bottom of the valley). The values of partial derivatives with respect to t and s are “relatively reasonable” and we can perform a step of the steepest descent changing the values of t and s . After this step, we can be on a side of the valley again and hence it is necessary to recompute the values of c_i and so further until we converge close enough to a minimum.

3 Experimental results

We tried both approaches on the implementation of GNFS polynomial selection in Prague [1]. The results are in Table 1. We applied the algorithm on many polynomials measuring the average time it took and the average value of $I(F, S)$ obtained, where $I(F, S) = \ln \sqrt{FS}$. The degree of $P(x)$ was chosen so that the performance of the algorithms is optimal: if the degree of $P(x)$ is too high, the highest coefficients are set to zero anyway, only the computation is slower and less exact.

| degree of $f_m(x)$ | degree of $P(x)$ | classical algorithm | | improved algorithm | |
|-----------------------|---------------------|---------------------|----------|--------------------|----------|
| | | $I(F, S)$ | time | $I(F, S)$ | time |
| 4 | 0 | 37.52 | 52.4 ms | 37.49 | 20.4 ms |
| 5 | 1 | 44.47 | 144.6 ms | 43.94 | 125.6 ms |
| 6 | 2 | 52.34 | 215.5 ms | 51.29 | 225.2 ms |

Table 1: Experimental results for the steepest descent algorithms

We can interpret the results the following way: when there is only one coefficient of $P(x)$, the valleys are not so narrow, the classical steepest descent converges (and it is almost as good as the improved version). However it converges more slowly. The more there are coefficients of $P(x)$, the more problems occur and the more probable is for the classical algorithm to stop

long before reaching a minimum. It can even finish sooner than the improved one but with a much worse result.

References

- [1] M. Kechlibar, P. Jedlička, L. Perútka, J. Zvánovec: An implementation of the general number field sieve
- [2] T. Kleinjung: *On polynomial selection for the general number field sieve*, Math. of Comput. **75** (256), 2006, 2037–2047
- [3] A. K. Lenstra, H. W. Lenstra (eds): “The development of the number field sieve”, Lecture Notes in Math. **1554**, 1993, Springer-Verlag.
- [4] C. Monico: “GGNFS”,
<http://www.math.ttu.edu/~cmonico/software/ggnfs/index.html>
- [5] B. A. Murphy: *Polynomial Selection for the Number Field Sieve Integer Factorisation Algorithm*, Ph.D. thesis, The Australian National University, 1999
- [6] C. Pomerance: *A Tale of Two Sieves*, Notices of AMS, **43** (12), 1996, 1473–1485

Czech University of Life Sciences,
Department of Mathematics,
Faculty of Engineering,
Kamýcká 129, 165 21, Praha 6 – Suchbát, Czech Republic
e-mail: jedlickap@tf.czu.cz