Factoring Integers with Large-Prime Variations of the Quadratic Sieve

Henk Boender and Herman J. J. te Riele

CONTENTS

AMS Subject Classification: 11A51, 11Y05.

Keywords: factorization, multiple polynomial quadratic sieve, vector supercomputer, cluster of workstations.

This article is concerned with the large-prime variations of the multipolynomial quadratic sieve factorization method: the PMPQS (one large prime) and the PPMPQS (two). We present the results of many factorization runs with the PMPQS and PPMPQS on SGI workstations and on a Cray C90 vector computer. Experiments show that for our Cray C90 implementations PPMPQS beats PMPQS for numbers of more than 80 digits, and that this crossover point goes down with the amount of available central memory.

For PMPQS we give a formula to predict the total running time based on a short test run. The accuracy of the prediction is within 10% of the actual running time. For PPMPQS we do not have such a formula. Yet in order to provide measurements to help determining a good choice of the parameters in PPMPQS, we factored *many* numbers. In addition we give an experimental prediction formula for PPMPQS suitable if one wishes to factor many large numbers of about the same size.

1. INTRODUCTION

Let n be an odd positive integer to be factored and suppose that n is not a prime power. If we can find two integers X and Y such that

$$X^2 \equiv Y^2 \bmod n, \tag{1.1}$$

then the greatest common divisor of $X - Y$ and n is a nontrivial factor of n if $X \not\equiv \pm Y \pmod{n}$. If X and Y are randomly chosen subject to (1.1), then this yields a proper factor of n in at least 50% of the tries. This principle is the basis for the best known general factorization methods, namely, the multipolynomial quadratic sieve, or MPQS [Bressoud 1989; Pomerance 1985; Pomerance et al. 1988; Silverman 1987; te Riele et al. 1989], and the number field sieve, or NFS [Lenstra and Lenstra 1993].

1058-6458/96 $0.50 per page

In this paper we discuss and compare the *single large-prime variation* (PMPQS) and the *double large-prime variation* (PPMPQS) of MPQS. An introduction to each of these methods is given starting in Section 2. We factor many numbers ranging from 66 to 88 decimal digits, mainly with PPMPQS, on either SGI workstations or a Cray C90 vector computer.

PPMPQS is known to be faster than PMPQS "by approximately a factor of 2.5 for sufficiently large n" [Lenstra and Manasse 1994], but the crossover point depends heavily on the choice of the parameters in the two methods, the computer, the available memory, and the implementation. It is stated further in [Lenstra and Manasse 1994] that PPMPQS was found to be faster than PMPQS for numbers of at least 75 decimal digits, and that the speed-up factor of 2.5 was obtained for numbers of more than 90 digits. As a comparison, a 106-digit number was factored with PMPQS in about 140 mips years, and a 107-digit number with PPMPQS in about 60 mips years, both with a factor base size of 65,500. A 116-digit number was factored with PPMPQS in about 400 mips years, with a factor base size of 120,000. No actual results for smaller numbers were given. Thomas Denny reports in his Master's Thesis [1993] various experiments with PPMPQS for numbers in the range of 75–95 decimal digits. From these experiments it is not clear where the crossover point for Denny's implementation lies. The largest numbers presently factored with PPMPQS are a 120-digit number done in about 825 mips years [Denny et al. 1994], and the 129-digit RSA challenge described by Martin Gardner, done in about 5000 mips years with a factor base size of 524,339 [Atkins et al. 1995].

A theoretical and practical problem with PPMPQS is the determination of the optimal parameters for a number of a given size. Since it only pays to use PPMPQS for rather large numbers, and since it is difficult to accurately predict the total running time of PPMPQS on the basis of a short test run (as contrasted with PMPQS), the precise effect of one specific choice of the parameters can only be measured accurately by carrying out the complete sieve part of the job. So in order to find the *optimal* parameter choice for a given number, that would minimize the CPU time, one would have to *repeat* the complete sieve job for several (10, say) different choices of the parameters. Of course, this does not make much sense since *one* sieve job will do to factor the number, so we decided to adopt the strategy to factor as many as possible *different* numbers in a not too wide decimal digits range, thus providing extensive experience with PPMPQS for many different numbers on the one hand, and contributing to a table of unfactored numbers [Brent and te Riele 1992] on the other hand. The price to pay for this strategy is that we can only give an *indication* of the optimal parameter choice for PPMPQS for numbers in the 65–90 decimal digits range.

We have implemented PPMPQS on an SGI workstation, and on a Cray vector computer. Some comparative experiments with PMPQS and PPMPQS on a Cray C90 indicated that for our implementation on that machine the crossover point lies around numbers having 80–85 decimal digits. For several different choices of the parameters in PPMPQS, we have factored eight numbers in the 66–83 digit range on an SGI workstation, and more than 70 numbers in the 67–88 digit range on a Cray C90 vector computer, as a contribution to the table in [Brent and te Riele 1992]. Most of these numbers were already tried before with the elliptic curve method (ECM), without success.

Section 2 discusses Dixon's algorithm. MPQS is described in Section 3. In Section 4 we treat the efficient generation of the polynomials in MPQS. In Section 5 the single large-prime variation of MPQS (PMPQS) is described. A known theoretical formula is worked out that helps to predict the total sieve time on the basis of a short test run. In this test run (of a few minutes CPU time, say) the speed is determined by which so-called complete and partial relations are generated during the sieve step of the algorithm; this speed is approximately constant during the whole sieve step. The accuracy of the prediction formula is within 10% of the actual sieve time. In Section 6 the double large-prime variation of MPQS is described, and an experimental prediction formula is given that has a restricted scope in the sense that it only applies to numbers of roughly the same size, and for a fixed choice of the parameters of the algorithm. In addition, for one particular number of 80 decimal digits, we have determined the optimal choice of one (of the four) parameters in PPMPQS as an illustration of the fact that this optimum is attained for a rather wide range of this parameter. Section 7 covers implementation aspects and discusses our experiments, including a comparison of our PMPQS- and PPMPQS-implementations for 71-, 87-, and 99-digit numbers. The paper closes with data from 81 factorizations.

2. THE BASIC IDEA

The algorithm described now is due to Dixon, who based it on the continued fraction method of Morrison and Brillhart [1975]. It is not efficient in practice compared to almost any other method, but it shows clearly the idea behind finding X and Y. So we mention it mainly for didactical reasons.

For $x \in \mathbb{Z}$ such that $|x| > \sqrt{n}$, define

$$g(x) :\equiv x^2 \bmod n.$$

(The notation $\sqrt{n}$ means the positive square root of n.) Suppose that we have a finite subset $J \subset \mathbb{Z}$ such that $\prod_{x \in J} g(x)$ is a square. Then we can take

$$X = \sqrt{\prod_{x\in J} g(x)}, \qquad Y = \prod_{x \in J} x.$$

A problem is how to determine J.

Choose a positive integer B_1, let $\pi = \pi(B_1)$ be the number of primes $\leq B_1$, and let $\{p_1, p_2, \ldots, p_\pi\}$ be the set of such primes. Suppose that we have a set T of $t > \pi$ numbers $g(x)$ only composed of primes $\leq B_1$, that is,

$$g(x) = p_1^{e_1(x)} p_2^{e_2(x)} \ldots p_\pi^{e_\pi(x)},$$

where $e_i(x)$ is the exponent of p_i in $g(x)$. Then

$$\prod_{x \in J} g(x) = \prod_{i=1}^{\pi} p_i^{\sum_{x\in J} e_i(x)}.$$

This is a square if and only if

$$\sum_{x \in J} e_i(x) \equiv 0 \bmod 2, \quad \text{for } i = 1, 2, \ldots, \pi.$$

Since $|T| = t > \pi$, there exists a nontrivial solution of this linear system of equations over GF(2). A solution can be found using Gaussian elimination. This yields at least $t - \pi$ useful subsets J.

3. THE MULTIPOLYNOMIAL QUADRATIC SIEVE

Dixon's algorithm does not tell us how to find T efficiently. Building on previous work of Kraitchik [1929], Lehmer and Powers [1931], Morrison and Brillhart [1975], and Schroeppel, Carl Pomerance [1982] introduced the *quadratic sieve algorithm*. It works with the quadratic polynomial

$$g(x) = (x + \lfloor \sqrt{n} \rfloor)^2 - n,$$

where x runs over the integers in $(-n^\epsilon, n^\epsilon)$, so that $g(x) = O(n^{1/2+\epsilon})$. With this $g(x)$ the set T may be built up, where some of the numbers $g(x)$ can be factored completely by a cheap sieve process because $g(x)$ is a polynomial (this is much more efficient than trial division or any other factoring method). We could also use a sieve process in Dixon's algorithm if we choose random numbers x in an arithmetic progression like $x, x+1, x+2, \ldots$. However, in practice this single polynomial $g(x)$ (or an arithmetic progression in Dixon's algorithm) does not give rise to a sufficiently large set T (with $t > \pi$ elements) in a reasonable amount of time. The reason for this is that the interval $(-n^\epsilon, n^\epsilon)$ is large when n is large and, since $g(x) = O(n^{1/2+\epsilon})$ (which is large), most numbers $g(x)$ are not likely to factor over a set of small primes. P. L. Montgomery found an efficient way to use *several* polynomials (thus introducing a simple way to run the algorithm *in parallel*), so that the numbers x can be taken from much smaller intervals rather than from one single very large interval. The average polynomial values then are smaller than the average value of g and are thus more likely to factor over small primes than the $g(x)$-values. If all the numbers in a small interval have been considered, we can pass to a next polynomial and try again. We describe here the resulting *multipolynomial quadratic sieve* method. We remark that Davis and Holdridge [1983] already had a multipolynomial version before Montgomery came up with his new idea. In fact, Montgomery's method is based on that of Davis and Holdridge, but it is slightly more efficient.

Suppose that we have integer numbers x, $U(x)$, $V(x)$, and $W(x)$ such that

$$U^2(x) \equiv V^2(x)W(x) \bmod n \quad \text{for all } x \in \mathbb{Z}. \tag{3.1}$$

If $J \subset \mathbb{Z}$ is a finite subset such that $\prod_{x\in J} W(x)$ is a square, we can take

$$X = \prod_{x\in J} V(x)\sqrt{\prod_{x\in J} W(x)}, \qquad Y = \prod_{x\in J} U(x).$$

In practice we choose $U(x) = a^2x + b$, $V(x) = a$, and $W(x) = a^2x^2+2bx+c$, with $|x| \le M$ (where M is a parameter we choose beforehand) and where a, b and c are integers that satisfy the following conditions [Bressoud 1989, p. 117]:

$$a^2 \approx \sqrt{2n}/M, \tag{3.2}$$

$$b^2 - n = a^2c, \tag{3.3}$$

$$|b| < a^2/2. \tag{3.4}$$

In the next section we describe how a, b and c are to be calculated.

$W(x)$ plays the role of $g(x)$ in Dixon's algorithm. In order to determine the subset J, we choose an upper bound B_1 for the primes. We want to have many $W(x)$-values that consist of primes $\le B_1$. However, only roughly half of the primes below B_1 can occur as a prime divisor of $W(x)$. Namely, if a prime p divides $W(x)$, then $p \mid a^2W(x)$ and thus $p \mid (a^2x+b)^2 - n$, which means that n is a quadratic residue modulo p. This leads to the definition of the *factor base* $\mathcal{F}$:

$$\mathcal{F} = \bigl\{\text{prime powers } q = p^k \le B_1 : \bigl(\tfrac{n}{p}\bigr) = 1\bigr\}.$$

(Of course, a prime can divide $W(x)$ more than once, so we also have to account for prime powers.) Note that $\mathcal{F}$ is independent of the choices of a, b and c, so we can use the same factor base for every proper choice of a, b and c.

Since $W(x)$ is more likely to be divisible by small primes than by large primes, it is advantageous that the factor base contains many small primes. We can construct such a factor base by multiplying the number n to be factored by a suitable small integer m, called the *multiplier*, and factoring mn rather than n [Pomerance et al. 1988, p. 391].

For a given $q \in \mathcal{F}$ the values of x for which q divides $W(x)$ can be found as follows. Compute the solution $t = t_q$ of the congruence equation

$$t^2 \equiv n \bmod q, \quad \text{for } 0 < t \le q/2$$

[Riesel 1985, pp. 212 and 287–288]. This has to be done only once during the algorithm. Now, if $q \mid W(x_0)$, then $q \mid (a^2x_0 + b)^2 - n$ and thus

$$x_0 \equiv a^{-2}(\pm t_q - b) \bmod q, \tag{3.5}$$

provided that $\gcd(a, q) = 1$. This is guaranteed by the choice of a (see Section 4). For each proper choice of a we compute $a^{-2} \bmod q$ for all $q \in \mathcal{F}$. In the next section we describe how these computations can be done. Furthermore, since $W(x)$ is a quadratic polynomial, q divides $W(x_0 + lq)$, for $l \in \mathbb{Z}$. So we can calculate efficiently the places where an element of $\mathcal{F}$ divides the W-values. This idea originated from Schroeppel.

Define the *report threshold* RT as the average of $\log|W(x)|$ on the interval $[-M, M]$, which is approximately $\log(\frac{1}{2}M\sqrt{n/2})$. Initialize a *sieve array* $\mathrm{SI}(-M, M)$ to zero and *sieve* with each $q = p^k \in \mathcal{F}$, i.e., add $\log p$ to $\mathrm{SI}(x_0 + lq)$ for all $l \in \mathbb{Z}$ such that $x_0 + lq$ is in the interval $[-M, M]$. For those numbers x for which $\mathrm{SI}(x) \ge \mathrm{RT}$, $W(x)$ is a good candidate for fully factoring over the factor base. In general, the time spent on sieving takes more than 85% of the total computing time.

Since sieving with small primes is expensive, it is customary *not* to sieve with the primes and prime powers $\leq$ QT, where QT is some suitably chosen threshold value. In order not to lose $W(x)$-values divisible by such small primes, the report threshold RT will be lowered by the amount $\sum_{p^k \leq \mathrm{QT}} \log p$. After the sieve step and the selection of those x for which $\mathrm{SI}(x) \geq \mathrm{RT}$, the prime factors of the corresponding $W(x)$ are found by comparison, for all $q \in \mathcal{F}$, of x with the two values of x_0 in (3.5) (which are computed and stored after the factor base has been computed). In this way, $W(x)$-values divisible by one or more of the small primes omitted during the sieving phase are not lost. If QT is suitably chosen, this can save a considerable amount of sieve time. This refinement of MPQS is known as the *small-prime variation.*

4. EFFICIENT CALCULATION OF THE POLYNOMIALS

Choose integers r and k such that $1 < k < r$ (typical choices are $r = 30$ and $k = 3$, for example). Generate primes $g_1, g_2, \ldots, g_r$, the so-called *g-primes*, such that (i) $g_i \approx (\sqrt{2n}/M)^{1/(2k)}$, (ii) $\left(\frac{n}{g_i}\right) = 1$, and (iii) $\gcd(g_i, q) = 1$, for $i = 1, 2, \ldots, r$ and for all $q \in \mathcal{F}$. Let

$$a = g_{i_1} g_{i_2} \cdots g_{i_k},$$

be the product of k g-primes, with $1 \leq i_1 < i_2 < \cdots < i_k \leq r$. Because of (i), this a satisfies condition (3.2).

Let b_i be a solution of the congruence equation

$$t^2 \equiv n \bmod g_i^2,$$

where $i = 1, 2, \ldots, r$. Solve the system of congruence equations (for a specific choice of the signs)

$$\begin{aligned} x &\equiv b_{i_1} \bmod g_{i_1}^2, \\ x &\equiv \pm b_{i_2} \bmod g_{i_2}^2, \\ &\vdots \\ x &\equiv \pm b_{i_k} \bmod g_{i_k}^2, \end{aligned} \tag{4.1}$$

by means of the Chinese Remainder Theorem. Let b be the solution of this system of equations. Then $b^2 \equiv n \bmod a^2$, so that condition (3.3) holds with

$$c = (b^2 - n)/a^2.$$

If $b \geq a^2/2$, we replace b by $b - a^2$ in order to satisfy condition (3.4). Since there are 2^{k-1} possible combinations of signs in (4.1), the number of polynomials that can be calculated with one set of r g-primes and a fixed k is $2^{k-1}\binom{r}{k}$.

If a new a has to be chosen, new sieve numbers x_0 subject to (3.5) must be computed. Since $a = g_{i_1} g_{i_2} \cdots g_{i_k}$, we can use

$$a^{-2} \bmod q = g_{i_1}^{-2} g_{i_2}^{-2} \cdots g_{i_k}^{-2} \bmod q.$$

Therefore, with the generation of the g-primes we also compute and store the numbers $g_i^{-2} \bmod q$, where $i = 1, 2, \ldots, r$, for all the prime powers q in the factor base.

For a fixed a, Alford and Pomerance [1995] developed a method to compute iteratively all the other values of b (and thus c) from a given initial value of b (see also [Peralta $\geq$ 1996]). They also pointed out how the two solutions in the interval $[0, q)$ of the congruence equation $W(x) \equiv 0 \bmod q$ can be calculated from the zeros mod q of a "previous" polynomial. With this improvement we obtain the *self-initializing variation* of MPQS. It has the advantage that it can change polynomials cheaply, so a shorter sieve interval can be used.

We have implemented this variation on an SGI workstation and on a Cray C90 vector computer. Some speed-up was observed on an SGI workstation when we reduced the length of the sieve interval, but other effects like an increasing loop overhead in the sieving step interfere with this in the opposite direction.

On a vector computer such as the Cray C90, reducing the length of the sieve interval reduces the vector lengths in the sieving step and, consequently, the efficiency of the vectorization. Therefore, we decided not to use the self-initializing variation of the quadratic sieve in our experiments.

5. THE LARGE-PRIME VARIATION OF MPQS

The *large-prime variation* of MPQS incorporates the following improvement, which is based on a step in the continued fraction algorithm of Morrison and Brillhart [1975]. $W(x)$ is allowed to have a factor $R > B_1$ that is not composed of primes from the factor base. If the cofactor R (after dividing out all factor base primes in $W(x)$) is less than or equal to B_1^2, it must be a prime. In order to restrict the amount of disk space needed for storage of the relations (3.1), we only accept factors $R \le B_2$, where B_2 is a parameter we choose beforehand. In practice we choose B_2 in such a way that B_2/B_1 is a number between 10 and 100. We have to lower the report threshold by $\log(B_2)$ in order to find these $W(x)$-values after sieving.

If we have found two $W(x)$-values with the same R, multiplication of the corresponding relations (3.1) yields a relation of the form (3.1), where $W(x)$ only consists of prime powers $q \in \mathcal{F}$ (and R is moved to $V(x)$).

A relation of the form (3.1), where $W(x)$ only consists of primes $q \in \mathcal{F}$, is called a *complete relation*. If $W(x)$ has one prime factor $R \le B_2$ (and the others are in $\mathcal{F}$), then the relation is called a *partial relation*.

We wish to compute E, the expected number of complete relations coming from a given number of r partial relations. Let

$$\Omega = \{\text{primes } q : B_1 < q \le B_2,\ \left(\tfrac{n}{q}\right) = 1\}.$$

The elements of Ω are called *large primes*. Let P_q be the probability that a large prime q occurs in a partial relation. Lenstra and Manasse [1994] assume that

$$P_q \approx q^{-\alpha} \Big/ \sum_{p\in\Omega} p^{-\alpha} \tag{5.1}$$

for some positive constant $\alpha < 1$ that should be determined experimentally. They report that $\alpha \in [\frac{2}{3}, \frac{3}{4}]$ gives a reasonable fit with their experimental results. Denny [1993, pp. 44–49] takes $\alpha = 0.775$.

From [Lenstra and Manasse 1994] it follows that

$$E = r - \#\Omega + \sum_{q\in\Omega} (1 - P_q)^r.$$

We apply the binomial formula of Newton and use approximation (5.1) to find

$$E \approx \sum_{i=2}^{r} (-1)^i \binom{r}{i} \Big(\sum_{q\in\Omega} q^{-\alpha}\Big)^{-i} \sum_{q\in\Omega} q^{-\alpha i}. \tag{5.2}$$

Since $\pi(t) \sim t/\log t$ as $t \to \infty$, we have

$$\sum_{p\le x} p^{-u} \approx \int_2^x t^{-u}\, d(t/\log t)$$

with p prime, $x \in \mathbb{R}_{\ge 2}$, $u \in \mathbb{R}_{>0}$. Hence for $u > 0$ we have

$$\sum_{q\in\Omega} q^{-u} \approx \frac{1}{2} \int_{B_1}^{B_2} t^{-u}\, d(t/\log t).$$

To compute the last integral we first use partial integration and then substitute $s = (1-u)\log t$. We get

$$\int_{B_1}^{B_2} t^{-u}\, d(t/\log t) = B_2^{1-u}/\log B_2 - B_1^{1-u}/\log B_1 + u\{\mathrm{Ei}((1-u)\log B_2) - \mathrm{Ei}((1-u)\log B_1)\},$$

where $\mathrm{Ei}(x) = \int_{-\infty}^{x} (e^s/s)\, ds$ is the exponential integral. Now combine the last three displayed equations for the appropriate choices of u to get an approximation for E. In approximation (5.2) we sum from $i = 2$ to $i = 5$ and forget about the higher-order terms to get a formula for an approximation of E that we can use in practice (given B_1, B_2, r, and α).

The experiments summarized in Table 1 show that our approximation works well if $\alpha = 0.73$. The table shows, for each example run, the number r of partial relations, the actual number of complete relations derived from these partial relations, and the estimated number of complete relations. An approximation of E can be used to *predict* the computing time.

n	$B_1/10^4$	B_2/B_1	r	actual	estim.
C75	30	20.0	37472	4790	4966
C80	10	60.0	15918	1121	1209
C80	30	167	68195	4113	4150
C84	80	25.0	96138	10894	11148
C88	50	100	94651	6605	6736
C88	75	100	148403	11455	11211
C88	75	100	158214	12830	12657
C88	75	100	146983	11051	11008
C88	75	100	150327	11498	11488
C88	70	100	148016	12116	11827

TABLE 1. For ten composite numbers and bounds B_1, B_2, we list the number r of partial relations, and the actual and estimated number of complete relations (last two columns). As usual, Cx denotes a composite number with x decimal digits.

To determine the best value of α, we wrote a program in Maple that, given α, computes the absolute value of the difference of the actual number of complete relations and the estimated number of complete relations for each of fifteen test numbers. Then we summed the fifteen absolute values of the differences, thus obtaining for each α a sum of absolute values. It turned out that $\alpha = 0.73$ gave rise to the smallest sum.

6. THE DOUBLE LARGE-PRIME VARIATION

In the large-prime variation of MPQS we allow $W(x)$ in (3.1) to have a prime factor R with $B_1 < R \le B_2$. In the double large-prime variation of MPQS we also let $W(x)$ have a factor $R \le B_2^2$ composed of *two* primes $> B_1$. In this case we call such a relation a *partial-partial* relation (pp-relation for short). Now the problem of finding combinations of partial and partial-partial relations that yield a *complete* relation can be formulated as finding cycles in an undirected graph: the vertices are the large primes and two vertices (primes) are connected by an edge if there is a pp-relation in which both primes occur. A partial relation is represented by adding 1 as a vertex to the graph. We consider this partial relation as a pp-relation where one of the large primes is 1. So an edge in the graph corresponds to a partial or partial-partial relation and a cycle corresponds to a set of relations with the following property: if we multiply these relations, then all the large primes in the product occur to an even power. Hence, for the linear algebra step this set can be viewed as a complete relation. To avoid dependent relations one only has to find the *basic* cycles of the graph.

The number of complete relations coming from the pp-relations is much more difficult to predict than that coming from the partial relations. One has to know how the number of basic cycles in a graph with given vertices varies when edges are added more or less randomly. Having a basic cycle is a monotone increasing property [Bollobás 1985, p. 33] that can appear rather suddenly [Erdős and Rényi 1959; 1960; 1961]. An algorithm for finding the basic cycles in a graph can be found in [Paton 1969].

If R is prime then we require $R < B_2$ in order to restrict the total number of relations (in our experience partial relations with $B_2 \le R < B_1^2$ do not contribute much to the total number of complete relations). If R is composite, its large prime factors can be found, e.g., by using Shanks' SQUFOF algorithm [Riesel 1985, pp. 191–199]. This algorithm has the advantage that most numbers that occur during its execution are in absolute value not larger than $2\sqrt{R}$.

We want to estimate the time that PPMPQS spends on the sieve step for numbers n of about d decimal digits, given B_1, M, B_2, and QT. To that end, let

$$\begin{aligned} n_f &= \text{number of elements in the factor base,} \\ n_c &= \text{number of complete relations,} \\ f_1 &= n_c/n_f, \\ n_1 &= \text{number of partial relations,} \\ n_2 &= \text{number of pp-relations,} \\ f_2 &= n_2/n_1, \\ T_s &= \text{sieve time.} \end{aligned}$$

During the sieve step, the numbers n_c, n_1 and n_2 grow (more or less) linearly with the time, so that

#	33	35	37	44	42	38	48	40	47	34	39	36	46	41	32	43
m	109	37	1	109	1	109	5	29	1	1	1	7	43	1	1	41
f_1	0.243	0.244	0.255	0.269	0.275	0.297	0.301	0.310	0.320	0.325	0.331	0.346	0.348	0.349	0.352	0.363
f_2	5.98	5.79	4.04	3.68	2.75	2.37	2.13	2.29	1.70	1.64	1.14	0.906	0.961	0.862	0.760	0.798

TABLE 2. Values of f_1 and f_2 measured for 16 numbers n from Table 7 (identified by the number in the first row). We used $d = 86$, $B_1 = 5 \times 10^5$, $M = 1.5 \times 10^6$, $B_2/B_1 = 20$, and QT = 40, with multiplier m.

also the fraction f_1 grows linearly, and f_2 stays more or less constant (after the sieve step has been running for a short time). We observed that the values of the fractions f_1 and f_2, measured after completion of the sieve step, seem to be connected; see Table 2.

The table suggests that f_2 is an exponential function of f_1, that is,

$$f_2 = ae^{bf_1}$$

for some constants a and b. Based on the table, we estimated $a = 315$ and $b = -16.5$. Since $\log f_2 = \log a + bf_1$, it follows that

$$n_c = \frac{1}{b}(\log f_2 - \log a) \cdot n_f.$$

If u is the time needed to generate one complete relation, we obtain the following approximation for the sieve time T_s:

$$T_s \approx (0.349 - 0.061 \log f_2) \cdot u \cdot n_f. \qquad (6.1)$$

We can estimate u and f_2 by letting the program run for a short while, five minutes say. The measurements shown in Table 3, pertaining to runs on the Cray C90 of several 85- and 86-digit numbers, suggest that the estimate works well.

#	m	u	f_2	n_f	T_s	approx.
21	19	5.140 s	1.1945	20741	9.8 h	10.0 h
22	1	4.518 s	0.7646	20744	9.8 h	9.50 h
24	1	3.357 s	1.4378	20930	6.0 h	6.37 h
31	1	4.226 s	1.0866	24641	10.0 h	9.94 h
48	5	8.785 s	2.1364	20911	15.4 h	15.4 h

TABLE 3. Tests of approximation (6.1). For five composite numbers from Table 7 (identified by the number in the first column), we measured the actual value of T_s and computed the value predicted by the approximation (last column).

Consequently, approximation (6.1) can be used to obtain a good estimate of T_s in the PPMPQS algorithm for numbers of about the same size, and fixed parameters B_1, M, B_2, and QT. For numbers in another range, or if we wish to change the parameters, some experiments have to be done to determine the total sieve time under these new conditions, by which the coefficients in (6.1) can be estimated.

In order to test the dependency of T_s on B_2, we carried out on the Cray C90 the *complete* sieve step of PPMPQS for the 80-digit number

$$\frac{75^{64}+1}{2 \cdot 224914177 \cdot 1511139087864219170368069437233933}, \qquad (6.2)$$

which has the two prime factors

687990387865123193888213509255 69 and
215768091527974049646247615957101365677594246657.

We kept $B_1 = 10^5$, $M = 3 \times 10^6$, and QT = 50 fixed, and tried various values of B_2. The statistics are shown in Table 4.

In the partial relations we allowed the large prime R to be less than B_1^2. (We get these relations free, because $R < B_1^2$ implies that R is prime.) For $B_1 = 10^5$ the number of elements in the factor base is 4806. The sieving was continued until the total number of complete relations, including those generated by the partial relations and the partial-partial relations, surpassed this number. We only measure the total number of complete relations obtained so far at selected points in our program, so the *actual* total number of complete relations is

B_2/B_1	T_s	n_c	n_1	$n_{c,1}$	n_2	$n_{c,2}$	total
30	8.64 h	1036	129318	1661	29143	2121	4818
60	7.06 h	871	117532	1249	51929	2739	4859
100	6.49 h	775	109506	1025	76324	3070	4870
200	6.02 h	685	99474	795	123001	3339	4819
400	5.67 h	618	91332	634	193278	3598	4850
600	5.71 h	578	87265	568	243015	3698	4844
800	5.62 h	563	84926	531	291177	3766	4869
1000	5.75 h	546	83082	501	333726	3796	4843
1600	6.19 h	521	79960	464	445526	3860	4845

TABLE 4. Number of relations as a function of B_2, for the factorization of (6.2) with $B_1 = 10^5$, $M = 3 \times 10^6$, and QT $= 50$. The column $n_{c,1}$ is the number of complete relations generated by the n_1 partial relations, and $n_{c,2}$ is the number of complete relations generated by combining the partial relations (with different large primes) and the n_2 pp-relations. "Total" is the sum $n_c + n_{c,1} + n_{c,2}$.

usually somewhat larger than the number of elements in the factor base.

As we increase B_2/B_1, the program generates more partial-partial relations and less complete and less partial relations in a given amount of sieve time. For $30 \leq B_2/B_1 \leq 400$, the gain in complete relations ($n_{c,2}$) generated by the pp-relations (n_2) more than sufficiently compensates for the loss of complete relations directly found by the sieve (n_c) and the loss of complete relations ($n_{c,1}$) generated by the partial relations (n_1). As a result, the total sieve time T_s *goes down.* For $B_2/B_1 > 1000$, however, the increase in size of the large primes in the partial and partial-partial relations is responsible for a decrease in the number of complete relations derived from these relations, and also the time that SQUFOF needs to find the two large primes in a pp-relation increases, so now the resulting total sieve time *increases.* Consequently, the minimal sieve time is reached if we choose B_2/B_1 in the interval $400 < B_2/B_1 < 1000$. In that interval the total sieve time is only slightly varying. We conclude that, in order also to minimize the amount of *memory* for storage of the relations, the optimal choice of B_2/B_1 is about 400.

7. IMPLEMENTATION AND EXPERIMENTS

For our PMPQS-experiments we used the implementation described in [te Riele et al. 1989]. Almost all our subroutines are written in Fortran.

We originally implemented the PPMPQS algorithm on a supercomputer like the Cray C90 vector computer. We used the same implementation on Silicon Graphics workstations. (We now have written a program especially designed for workstations).

The sieve operations (i.e., additions of $\log p$ to an element of the sieve array) are done in 64-bits floating-point arithmetic on Cray and in 32-bits on SGI. The maximum speed we obtained (in millions of sieve operations per second) was 3.3 on the Silicon Graphics, 110 on the Cray Y-MP [te Riele et al. 1991] and 270 on the Cray C90. The maximum speed was 5.7 when we used the workstation version of our program.

We used a package of Winter in order to carry out multiprecision integer arithmetic.

The large prime R occurring in the partial relations was accepted if $B_1 < R < B_2$ and rejected if $B_2 \leq R < B_1^2$.

We have implemented Paton's cycle-finding algorithm [1969] and used it as a preprocessing step for the Gaussian elimination step in PPMPQS.

An algorithm for just *counting* (not finding) the basic cycles [Lenstra and Manasse 1994, pp. 789–790; Denny 1993, pp. 61–64] was implemented by us as a tool to check during the sieve part of PPMPQS whether sufficiently many relations (complete, partial, and partial-partial) were collected.

The method used to do the Gaussian elimination modulo 2 is described in [Parkinson and Wunderlich 1984]. The elements of the bit-array are packed in words of 64 bits (on the Cray computers) or 32 bits (on the Silicon Graphics). This allows the use of the exclusive-or operation with the column vectors of the array, which is very efficient. The total Gaussian elimination step (including finding basic cycles) accounts for less than 0.6% of the total work of the PPMPQS algorithm.

	B_1	n_f	B_2/B_1	M	PMPQS T_s	n_c	n_1	$n_{c,1}$	PPMPQS T_s	n_c	n_1	$n_{c,1}$	n_2	$n_{c,2}$
C71	3×10^5	12979	20	5.0×10^5	0.58 h	10204	17993	2784	0.55 h	5063	36468	4709	42617	3400
C71	6×10^5	24510	20	5.0×10^5	0.56 h	20827	23794	3703	0.96 h	10868	68019	8383	70395	5389
C71	6×10^5	24510	40	5.0×10^5	0.55 h	20312	30399	4209	1.28 h	9817	80017	7390	132290	7412
C71	6×10^5	24510	40	2.5×10^6	0.29 h	20196	31034	4359	1.21 h	9803	81612	7499	138147	7969
C80	10^5	4806	400	3.0×10^6	13.4 h	1580	49143	3229	5.67 h	618	91332	634	193278	3598
C87	5×10^5	20838	20	2.5×10^6	16.4 h	9902	70029	10940	11.9 h	7009	63089	8220	57513	5620

TABLE 5. Comparison of PMPQS and PPMPQS. The C71 and C87 are listed on this page, the C80 in (6.2) on page 264.

In order to compare PMPQS with PPMPQS we have run our implementations of these algorithms on the Cray C90 for the 71-digit number

$$C71 = (10^{71} - 1)/9$$

and for the 87-digit cofactor

C87 = 1360245925758378639396610479463908049304-
23542841197990430220444148923901462079070640121

of $72^{99} + 1$. For C71, four experiments with different combinations of B_1, B_2/B_1, and M were carried out where in the second, third and fourth experiment only one of the three parameters was changed compared with the previous experiment. The value of QT was kept fixed at 40. For C80 from (6.2), which was treated in the previous section with PPMPQS, we made a comparison run with PMPQS for $B_1 = 10^5$, $M = 3 \times 10^6$, QT = 50, and $B_2/B_1 = 400$ (the optimal choice for PPMPQS). The results are given in Table 5.

For C71, the parameter choice $B_1 = 3 \times 10^5$, $B_2/B_1 = 20$, and $M = 5 \times 10^5$ yields a somewhat smaller sieve time for PPMPQS (0.55 CPU hours) than for PMPQS (0.58), but if we allow more memory use by choosing $B_1 = 6\times10^5$ and $M = 2.5\times10^6$ (and $B_2/B_1 = 40$), then PMPQS beats PPMPQS (0.29 vs. 1.21). Increasing the length of the sieve interval (M from 5×10^5 to 2.5×10^6) particularly improves the efficiency of PMPQS (and, to a lesser extent, of PPMPQS). For C87, with the parameter choice $B_1 = 5 \times 10^5$, $B_2/B_1 = 20$, and $M = 2.5 \times 10^6$, PPMPQS is faster than PMPQS (11.9 vs. 16.4).

We conclude that for our implementations PPMPQS can beat PMPQS for numbers of more than 80 (say) decimal digits, but the crossover point strongly depends on the amount of available central memory. For practical reasons (like throughput) it can be profitable to reduce the size of a sieve job on the Cray C90, so even though such a computer has a very large central memory, it is still worthwhile to restrict the size of the upper bound on the primes in the factor base and to have an efficient implementation of a memory-economic method like PPMPQS. This is even more important on workstations, particularly when there are primary and secondary cache memories (as is usual on workstations).

Furthermore, with our PMPQS program we have factored the 99-digit cofactor

1684830849783397621153043603997266025308430041776-
92574904043633682183896384221755952112008347771913

of the "more wanted" C133 with code 2,914M in the Cunningham table [Wagstaff 1993]. This C133 is the number $(2^{457} + 2^{229} + 1)/(5 \times 71293)$; Peter Montgomery had found the 34-digit prime factor

$$6196333979234679466021864314534473$$

with ECM, and left the 99-digit composite factor. We decomposed it into the product of the 49- and a 50-digit primes,

5845296257595668545524969937697507923682374822769 ×
28823703291241135239378075616078003806433692452377

with the help of an eight-processor IBM 9076 SP1, and 69 Silicon Graphics workstations (63 at CWI and 6 at Leiden University). The factor base size was 56976 with $B_1 = 1.5 \times 10^6$, $B_2/B_1 = 50$, $M = 2 \times 10^6$, and QT = 30. Parallel processing with good load balancing was effectuated by assigning different polynomials to different workstations. The total amount of sieve time was about 19,500 workstation CPU hours. The physical time for this factorization was about four weeks. This means that we consumed about 40% of the total CPU capacity of these workstations during that period (assuming that they all are equally fast: in fact, an RS 6000 processor of the IBM SP1 sieved about twice as fast as an SGI workstation). The Gaussian elimination step was carried out on a Cray C90; it required about 0.5 Gbytes of central memory, and one hour CPU time.

As a comparison with a vector computer [te Riele et al. 1991], on a Cray Y-MP we factored a 101-digit more wanted Cunningham number with PMPQS in 475 CPU hours, using $B_1 = 1300000$, with 50179 primes in the factor base, $B_2/B_1 = 50$, $M = 4.5 \times 10^6$, and QT = 40 (our PMPQS implementation runs about twice as fast on the Cray C90 as on the Cray Y-MP).

As a comparison with PPMPQS, from the results listed on the right in Table 5 we estimate (based on the assumption that the computing time of PPMPQS approximately doubles if the size of the number increases by three decimal digits) that we would roughly need 10,000 CPU hours of an SGI workstation to factor the 99-digit cofactor of 2,914M C133, yielding a speed-up factor of about 2 compared to PMPQS. If we would take a factor 1.64 (see the next paragraph) instead of 2, then the time would be less than 4000 CPU hours.

Tables 6 and 7, on pages 268–271, list the results of our experiments with PPMPQS on eight numbers in the 66–83 digit range on an SGI workstation, and 73 numbers in the 67–88 digit range on a Cray C90 vector computer. Most of these numbers fill gaps in the table found in [Brent and te Riele 1992], and are difficult to factor, having been tried before with ECM without success. The factorizations of some numbers of the form $a^n \pm 1$ that are outside the range covered by that reference are also given in Table 7.

We have varied the parameters B_1, B_2/B_1, and M on different numbers (but not in a very systematic way) and kept QT = 40 fixed. We observe that the average CPU time for numbers in the 67–88 digit range varies between 0.4 and 12 CPU hours, so that increasing the number of digits by three gives an increase of the sieve time by a factor of about 1.64. This is smaller than the factor of 2 that is usually observed for PMPQS.

ACKNOWLEDGEMENTS

We thank Arjen Lenstra, Walter Lioen and Rob Tijdeman for reading the paper and for suggesting several improvements. Walter Lioen helped us with the implementation of our programs on SGI workstations and on Cray vector computers. We gratefully acknowledge the Dutch National Computing Facilities Foundation NCF for the provision of computer time on Cray Y-MP and Cray C90 vector processors. Finally, we acknowledge the help of IBM and the Academic Computing Center Amsterdam (SARA) for providing access to and CPU time on the IBM SP1 at SARA.

REFERENCES

[Alford and Pomerance 1995] W. R. Alford and C. Pomerance, "Implementing the self-initializing quadratic sieve on a distributed network", pp. 163–174 in *Number-theoretic and algebraic methods in computer science* (NTAMCS '93: Moscow, 1993), edited by A. J. van der Poorten et al., World Scientific, River Edge, NJ, 1995.

[Atkins et al. 1995] D. Atkins, M. Graff, A. K. Lenstra, and P. C. Leyland, "The magic words are squeamish ossifrage", pp. 263–277 in *Advances in cryptology* (ASIACRYPT '94: Wollongong, 1994), edited by J. Pieprzyk and R. Safavi-Naini, Lecture Notes in Comp. Sci. **917**, Springer, Berlin, 1995.

#	n	prime factor(s)
1	C66 from $77^{53}+1 = \text{P31}\cdot\text{P35}$	P31 = 8508101816450689975658227843439
2	C67 from $58^{88}+1 = \text{P26}\cdot\text{P41}$	P26 = 62057338333442627487392257
3	C67 from $62^{89}-1 = \text{P31}\cdot\text{P37}$	P31 = 3916898265747514256035560079891
4	C75 from $70^{87}+1 = \text{P29}\cdot\text{P46}$	P29 = 56476537654063551106920429541
5	C79 from $72^{118}+1 = \text{P38}\cdot\text{P42}$	P38 = 16059490907009321225480347480687832441
6	C82 from $84^{71}+1 = \text{P33}\cdot\text{P50}$	P33 = 133184106044570646620234096956423
7	C82 from $80^{99}+1 = \text{P32}\cdot\text{P51}$	P32 = 11935171798229644025656192643827
8	C83 from $92^{87}+1 = \text{P23}\cdot\text{P61}$	P23 = 10127992394070979564027

TABLE 6. Parameter choices, timings, and factors for numbers ranging from 66 to 83 decimal digits, factored with PPMPQS on a SGI workstation. Key: n = number to be factored ("Cx from y" means a composite factor of y having x decimal digits); $d = \log_{10} n$; B_1 = upper bound for the primes in the factor base; B_2^2 = upper bound for the input R to SQUFOF (yielding a pp-relation); n_f = number of primes in the factor base;

#	n	prime factor(s)
1	C67 from $89^{64}+1 = \text{P24}\cdot\text{P44}$	P24 = 153316525308739316934017
2	C69 from $50^{122}+1 = \text{P30}\cdot\text{P40}$	P30 = 276832194921994230575098974137
3	C75 from $101^{41}+1 = \text{P32}\cdot\text{P43}$	P32 = 21587227703328821952030527314507
4	C75 from $110^{41}+1 = \text{P16}\cdot\text{P25}\cdot\text{P35}$	P16 = 3850561614882023 P25 = 7797598239853074057655219
5	C75 from $110^{47}+1 = \text{P24}\cdot\text{P51}$	P24 = 728424414211828929294823
6	C75 from $35^{147}+1 = \text{P35}\cdot\text{P40}$	P35 = 86052439411099140168070862933143801
7	C75 from $53^{59}-1 = \text{P24}\cdot\text{P51}$	P24 = 943970114867362247759443
8	C78 from $19^{165}+1 = \text{P28}\cdot\text{P50}$	P28 = 2481953419044452308291386601
9	C78 from $51^{102}+1 = \text{P30}\cdot\text{P48}$	P30 = 459028910227193494771112394289
10	C80 from $86^{58}+1 = \text{P33}\cdot\text{P47}$	P33 = 129094951090723152084884804969621
11	C80 from $75^{64}+1 = \text{P32}\cdot\text{P48}$	P32 = 68799038786512319388821350925569
12	C80 from $59^{85}-1 = \text{P36}\cdot\text{P44}$	P36 = 192052183634195717382812875959337681
13	C80 from $76^{123}+1 = \text{P28}\cdot\text{P53}$	P28 = 1602475801546350975094860307
14	C80 from $84^{87}-1 = \text{P40}\cdot\text{P41}$	P40 = 2904043752413366850400636076474517615769
15	C81 from $18^{103}-1 = \text{P35}\cdot\text{P47}$	P35 = 15936754604932361311519937275763087
16	C83 from $82^{68}+1 = \text{P40}\cdot\text{P43}$	P40 = 9241855378580566956862595601843404638609
17	C83 from $93^{71}+1 = \text{P34}\cdot\text{P50}$	P34 = 1871598891695207952802939248474557
18	C84 from $89^{67}-1 = \text{P41}\cdot\text{P44}$	P41 = 17345460386856072657168883886351357651503
19	C84 from $74^{91}-1 = \text{P31}\cdot\text{P54}$	P31 = 6300454649733691099786120178647
20	C85 from $69^{117}+1 = \text{P42}\cdot\text{P43}$	P42 = 553775456930001686459646662784000439421893
21	C85 from $98^{91}+1 = \text{P39}\cdot\text{P47}$	P39 = 150856027763097994901861400756223948651
22	C85 from $80^{58}+1 = \text{P42}\cdot\text{P44}$	P42 = 587407531780545617292693056474932755332969
23	C85 from $56^{64}+1 = \text{P43}\cdot\text{P43}$	P43 = 1120971223480359091305712645673434758493441
24	C85 from $39^{111}-1 = \text{P32}\cdot\text{P54}$	P32 = 38661901037861787717347412050407
25	C85 from $77^{95}-1 = \text{P34}\cdot\text{P52}$	P34 = 1254200040785197567017611121581711
26	C86 from $18^{111}+1 = \text{P35}\cdot\text{P51}$	P35 = 57095169829153516132919139336069139
27	C86 from $76^{59}+1 = \text{P39}\cdot\text{P47}$	P39 = 471586815074704431240140019672222092489
28	C86 from $20^{97}+1 = \text{P34}\cdot\text{P52}$	P34 = 2645332912014287669339495089951567

TABLE 7. Parameter choices, timings, and factorizations for numbers ranging from 67 to 88 decimal digits,

#	d	$B_1/10^5$	n_f	B_2/B_1	$M/10^5$	n_c	n_1	$n_{c,1}$	n_2	$n_{c,2}$	T_s
1	65.56	0.8	3911	11.25	2	1493	9753	1715	4102	710	5.8 h
2	66.17	0.8	3908	10	1.5	1452	9433	1766	3697	693	4.8 h
3	66.83	0.8	3984	10	2	1214	9952	2139	4238	637	14.2 h
4	74.15	3	13045	20	6	4840	37472	4790	26391	3424	55.4 h
5	78.76	3	12898	30	5	4444	44583	5104	29653	3355	123 h
6	81.54	5	20812	20	5	7992	63176	8471	33614	4351	173 h
7	81.70	4.5	18961	20	4.5	6796	55435	7229	38950	4942	198 h
8	82.89	5	20861	20	8	7387	62346	8229	40035	5250	273 h

$[-M, M]$ = sieve interval; n_c = number of complete relations found immediately; n_1 = number of partial relations; $n_{c,1}$ = number of complete relations coming from partial relations; n_2 = number of pp-relations; $n_{c,2}$ = number of complete relations coming from pp-relations; T_s = sieve CPU time. The small-prime variation parameter QT is always 40.

#	d	$B_1/10^5$	n_f	B_2/B_1	$M/10^5$	n_c	n_1	$n_{c,1}$	n_2	$n_{c,2}$	T_s
1	66.80	2	8881	30	25	2945	27673	2762	31855	3347	0.36 h
2	68.74	2.5	11086	20	5	3988	30107	3631	27746	3476	0.46 h
3	74.20	3.16	13623	20	6.31	4921	38371	4889	29855	3822	1.22 h
4	74.51	3.16	13625	20	6.31	5503	42284	5844	17604	2297	1.16 h
5	74.69	1	4790	60	5	1005	17630	1320	29502	2465	2.42 h
6	74.83	3	12892	17	25	4697	37137	5388	19447	2820	1.20 h
7	74.92	2.5	11086	36	25	3339	35899	3335	43531	4382	1.91 h
8	77.37	5	20972	20	25	7152	54706	6444	60361	7393	1.84 h
9	77.56	5	20888	30	30	7518	65930	6980	60042	6453	1.41 h
10	79.04	5	20597	30	30	6596	61563	6201	76295	7828	2.43 h
11	79.17	4	16927	20	1	5619	45717	5584	48399	6279	3.29 h
12	79.17	5	20895	20	3	6457	72272	11650	37114	2802	2.68 h
13	79.39	3	13001	166.7	3	3739	68195	4113	72708	5157	2.27 h
14	79.87	3	13011	166.7	3	3323	64308	3624	91150	6084	3.41 h
15	80.86	5	20819	20	6	6925	57619	7050	55281	6877	3.36 h
16	82.82	6	24598	20	2.5	8522	68723	8378	59901	7713	4.82 h
17	82.91	7	28413	20	2.5	11451	87010	11694	40636	5271	4.38 h
18	83.66	8	32104	25	2.5	11419	96138	10894	85260	9807	5.46 h
19	83.98	7	27980	25.7	2.5	10594	93766	11327	51233	6070	6.59 h
20	84.10	5	20713	20	2.5	6175	51592	5808	76377	8732	5.6 h
21	84.35	5	20741	20	2.5	6865	60444	7638	72201	6256	9.8 h
22	84.80	5	20744	20	2.5	7809	57576	7457	44022	5481	9.8 h
23	84.87	5	20790	20	2.5	7153	61546	7923	43044	5721	8.4 h
		5	20790	40	2.5	6412	73385	6960	75133	7427	8.4 h
24	84.92	5	20930	20	2.5	6434	52315	5865	75217	8614	6.0 h
25	84.99	5	20749	20	2.5	7106	58607	7259	53507	6389	6.8 h
26	85.02	5	20675	20	2.5	6982	61080	7920	64746	5774	9.8 h
27	85.02	5	20792	20	2.5	6679	58782	7268	81258	6853	11. h
28	85.05	5	20887	20	2.5	7754	65228	8990	46265	4178	8.4 h

factored with PPMPQS on a Cray C90 vector computer. Key as in Table 6. Continued overleaf.

#	n	prime factor(s)
29	C86 from $93^{99}-1=P31\cdot P55$	P31 = 3466732593888008254791613360081
30	C86 from $58^{93}-1=P32\cdot P54$	P32 = 75701865042739143157590250368211
31	C86 from $56^{96}+1=P39\cdot P47$	P39 = 232559086557407467762901333407938321409
32	C86 from $92^{84}+1=P43\cdot P43$	P43 = 2465152715658748428830880994824343639019833
33	C86 from $67^{99}-1=P34\cdot P52$	P34 = 2515208214206285121254951932641469
34	C86 from $13^{138}+1=P29\cdot P57$	P29 = 54836637716450236990971812089
35	C86 from $59^{89}-1=P31\cdot P55$	P31 = 2689941424488348023848649808389
36	C86 from $21^{123}+1=P39\cdot P47$	P39 = 380770063539669474313312691529545132713
37	C86 from $38^{81}-1=P36\cdot P50$	P36 = 511662075163970762060417538436484323
38	C86 from $31^{117}-1=P39\cdot P47$	P39 = 250630033376957433234617073114910871767
39	C86 from $50^{96}+1=P35\cdot P51$	P35 = 36774112300765382067961168652800897
40	C86 from $96^{95}+1=P28\cdot P58$	P28 = 2418476990688796014581890831
41	C86 from $24^{130}+1=P36\cdot P50$	P36 = 684989928644194001785075922656446841
42	C86 from $93^{53}+1=P38\cdot P49$	P38 = 19192699869550253389095978550167828173
43	C86 from $98^{59}+1=P32\cdot P55$	P32 = 29037047448209810589475647292291
44	C86 from $80^{65}+1=P31\cdot P55$	P31 = 3416871674919158699528742801241
45	C86 from $8^{2^7}+7^{2^7}=P42\cdot P44$	P42 = 519975935060346660783986052760977025136897 P44 = 65757674240355835167624181741955409969833473
46	C86 from $23^{83}-1=P38\cdot P49$	P38 = 27736074503263071062950778805992164759
47	C86 from $76^{56}+1=P40\cdot P47$	P40 = 4868699568817220592890920460964327586529
48	C86 from $47^{67}-1=P32\cdot P55$	P32 = 21270964162538089013014983761851
49	C86 from $67^{76}+1=P42\cdot P45$	P42 = 315618216027848486834301078445774290254513
50	C86 from $39^{81}+1=P37\cdot P50$	P37 = 2443003616566663069989278441133518059
51	C86 from $22^{95}-1=P34\cdot P52$	P34 = 9624357919068403555091512367414261
52	C86 from $76^{117}-1=P42\cdot P45$	P42 = 606202897105850025527074421945484005533987
53	C86 from $95^{80}+1=P38\cdot P49$	P38 = 45089758099791867831637486244759667041
54	C87 from $62^{65}+1=P34\cdot P53$	P34 = 1439106922902522842484110155444391
55	C87 from $72^{99}+1=P28\cdot P59$	P28 = 8097540789168990910686588841
56	C87 from $92^{85}-1=P32\cdot P56$	P32 = 14285278844357974752432939513571
57	C87 from $30^{95}+1=P35\cdot P52$	P35 = 80451911996934444483653727156040931
58	C87 from $50^{100}+1=P41\cdot P46$	P41 = 58951478878513071930500886762077392077601
59	C87 from $66^{96}+1=P42\cdot P46$	P42 = 153055732248039041786999207837459270270017
60	C87 from $19^{101}-1=P25\cdot P62$	P25 = 5245647644316863182854571
61	C87 from $33^{85}+1=P33\cdot P54$	P33 = 249536921989169261065035112257901
62	C87 from $63^{65}+1=P42\cdot P46$	P42 = 108410889974425685059575647391841055155451
63	C87 from $42^{99}-1=P33\cdot P55$	P33 = 234373090934137193434426100841739
64	C87 from $77^{67}-1=P41\cdot P46$	P41 = 75024943244844149373705126243013155715853
65	C87 from $84^{59}-1=P35\cdot P53$	P35 = 11779548019122302808328920808327631
66	C87 from $26^{129}+1=P31\cdot P57$	P31 = 3076814278757622588317626405309
67	C87 from $33^{111}-1=P38\cdot P50$	P38 = 21457939605898871224437297672972660829
68	C87 from $86^{84}+1=P40\cdot P48$	P40 = 1039512269081394539159468072656199331337
79	C87 from $85^{65}+1=P40\cdot P48$	P40 = 4645176624103101144238593467706089788481
70	C87 from $45^{85}+1=P36\cdot P52$	P36 = 218136090485068920975060625740020221
71	C87 from $87^{93}+1=P35\cdot P53$	P35 = 65234702723152738657728499902597613
72	C87 from $45^{71}+1=P27\cdot P61$	P27 = 692298161874034730813881603
73	C88 from $19^{168}+1=P42\cdot P47$	P42 = 261688712348581672325146786097393313497473

#	d	$B_1/10^5$	n_f	B_2/B_1	$M/10^5$	n_c	n_1	$n_{c,1}$	n_2	$n_{c,2}$	T_s
29	85.11	5	20810	20	2.5	4923	43182	4064	280566	11857	8.4 h
30	85.11	5	20841	20	2.5	5615	50651	5434	182705	9822	10.7 h
31	85.12	6	24641	20	2.5	8518	67320	9253	73153	6953	10.0 h
32	85.12	5	20651	20	1.5	7269	64239	8799	48843	4625	9.5 h
33	85.14	5	20812	20	1.5	5064	43981	4223	263194	11614	10.7 h
34	85.21	5	20709	20	1.5	6722	56788	6924	92891	7136	8.27 h
35	85.26	5	20859	20	1.5	5101	44412	4378	256996	11412	11.0 h
36	85.26	5	20768	20	1.5	7186	63721	8449	57739	5154	12.4 h
37	85.31	5	20812	20	1.5	5297	45852	4584	185169	10967	7.45 h
38	85.31	5	20576	20	1.5	6107	55044	6362	130553	8115	13.1 h
39	85.33	5	20709	20	1.5	6859	60552	7686	68840	6177	11.5 h
40	85.35	5	20923	20	1.5	6480	55476	6546	127090	7903	10.7 h
41	85.37	5	20672	20	1.5	7221	62980	8435	54345	5029	9.65 h
42	85.42	5	20672	20	1.5	5695	50790	5707	139604	9308	10.4 h
43	85.49	5	20772	20	1.5	7530	63927	8600	51034	4656	11.9 h
44	85.52	5	20634	20	1.5	5556	50383	5456	185347	9653	13.7 h
45	85.53	5	20711	20	2.5	5054	43759	4078	270308	11587	11.4 h
46	85.59	5	20797	20	1.5	7244	63668	8524	61191	5044	12.6 h
47	85.70	5	20712	20	1.5	6637	56910	6862	96694	7226	10.3 h
48	85.72	5	20911	20	1.5	6311	56349	6710	120384	7895	15.4 h
49	85.73	6	26392	2	3	16159	24514	9487	3153	749	13.8 h
50	85.92	6	26363	2	3	16376	24473	9358	7417	631	12.1 h
51	85.93	3	13041	20	1.5	4117	39602	5212	39395	3713	17.4 h
52	85.95	3	13011	20	2.5	4255	40517	5390	24478	3366	20.6 h
53	85.98	5	20756	2.4	2.5	10516	22044	7450	6610	2795	15.7 h
54	86.04	5	20840	20	2.5	7153	62231	8139	63273	5557	14.0 h
55	86.13	5	20838	20	2.5	7009	63089	8220	57513	5620	11.9 h
56	86.16	5	20787	22	2.5	7367	63987	8559	54708	4900	10.8 h
57	86.18	5	20688	20	2.5	7447	64778	8836	47154	4419	10.7 h
58	86.22	5	20852	20	2.5	6202	54180	6282	144069	8376	11.6 h
59	86.22	5	20947	20	2.5	7522	63620	8412	52191	5091	9.35 h
60	86.27	5	20978	40	2.5	6773	79489	8184	75416	6035	14.2 h
61	86.29	5	20797	40	2.5	6387	72868	6909	116000	7520	11.1 h
62	86.38	5	20754	40	2.5	6881	76861	7638	86915	6253	6.72 h
63	86.43	5	20920	40	2.5	7177	76854	7706	82085	6054	8.81 h
64	86.45	5	20631	40	2.5	6329	74485	7262	92362	7046	14.6 h
65	86.63	5	20902	80	2.5	5806	85167	6249	148784	8876	13.8 h
66	86.64	6	24404	100	3	7564	124510	9691	89594	7355	13.2 h
67	86.69	6	24573	100	3	7803	122935	9614	89375	7369	14.1 h
68	86.70	6	24495	100	3	6571	105037	7010	149698	11272	12.1 h
69	86.73	6	24538	100	3	7635	120888	9389	99930	7811	11.5 h
70	86.75	6	24374	100	3	7827	126178	9899	80444	6862	14.7 h
71	86.82	6	24615	100	3	6532	121187	9864	167590	8507	11.8 h
72	86.96	6	24658	100	3	7762	116023	8546	126334	8798	7.66 h
73	87.54	5	20604	100	1	6101	94651	6605	108893	8048	12.1 h

[Bollobás 1985] B. Bollobás, *Random graphs*, Academic Press, London and Orlando, FL, 1985.

[Brent and te Riele 1992] R. P. Brent and H. J. J. te Riele, "Factorizations of $a^n \pm 1$, $13 \leq a < 100$", Technical Report NM-R9212, Centrum voor Wiskunde en Informatica, Kruislaan 413, 1098 SJ Amsterdam, The Netherlands, June 1992. Updates to this report have appeared as CWI Report NM-R9419, September 1994 and CWI Report NM-R9609, March 1996, with P. L. Montgomery as a coauthor. The complete tables, incorporating these updates, are available at ftp://nimbus.anu.edu.au/pub/Brent/factors or ftp://ftp.cwi.nl/pub/herman/factors.

[Bressoud 1989] D. M. Bressoud, *Factorization and primality testing*, Undergraduate Texts in Math., Springer, New York, 1989.

[Davis and Holdridge 1983] J. A. Davis and D. B. Holdridge, "Factorization using the quadratic sieve algorithm", Technical Report Sand 83-1346, Sandia National Laboratories, Albuquerque, NM, 1983.

[Denny 1993] T. F. Denny, "Faktorisieren mit dem quadratischen Sieb", Master's thesis, Universität des Saarlandes, Saarbrücken, 1993.

[Denny et al. 1994] T. F. Denny, B. Dodson, A. K. Lenstra, and M. S. Manasse, "On the factorization of RSA-120", pp. 166–174 in *Advances in Cryptology* (CRYPTO '93: Santa Barbara, CA, 1993), edited by D. R. Stinson, Lecture Notes in Comp. Sci. **773**, Springer, Berlin, 1994.

[Erdős and Rényi 1959] P. Erdős and A. Rényi, "On random graphs I", *Publ. Math. Debrecen* **6** (1959), 290–297.

[Erdős and Rényi 1960] P. Erdős and A. Rényi, "On the evolution of random graphs", *Publ. Math. Inst. Hungar. Acad. Sci.* **5** (1960), 17–61.

[Erdős and Rényi 1961] P. Erdős and A. Rényi, "On the evolution of random graphs", *Bull. Inst. Int. Statist. Tokyo* **38** (1961), 343–347.

[Kraitchik 1929] M. Kraitchik, *Recherches sur la théorie des nombres*, II, Gauthier-Villars, Paris, 1929.

[Lehmer and Powers 1931] D. H. Lehmer and R. E. Powers, "On factoring large numbers", *Bull. Amer. Math. Soc.* **37** (1931), 770–776.

[Lenstra and Lenstra 1993] A. K. Lenstra and H. W. Lenstra, Jr. (editors), *The development of the number field sieve*, Lecture Notes in Math. **1554**, Springer, Berlin, 1993.

[Lenstra and Manasse 1994] A. K. Lenstra and M. S. Manasse, "Factoring with two large primes", *Math. Comp.* **63** (1994), 785–798.

[Morrison and Brillhart 1975] M. A. Morrison and J. Brillhart, "A method of factoring and the factorization of F_7", *Math. Comp.* **29** (1975), 183–205.

[Parkinson and Wunderlich 1984] D. Parkinson and W. Wunderlich, "A compact algorithm for Gaussian elimination over GF(2) implemented on highly parallel computers", *Parallel Comput.* **1** (1984), 65–73.

[Paton 1969] K. Paton, "An algorithm for finding a fundamental set of cycles of a graph", *Comm. ACM* **12** (1969), 514–518.

[Peralta $\geq$ 1997] R. Peralta, "Implementation of the hypercube variation of the multiple polynomial quadratic sieve". To appear.

[Pomerance 1982] C. Pomerance, "Analysis and comparison of some integer factoring algorithms", pp. 89–139 in *Computational methods in number theory*, I, edited by H. W. Lenstra, Jr. and R. Tijdeman, Mathematisch Centrum, Amsterdam, 1982.

[Pomerance 1985] C. Pomerance, "The quadratic sieve factoring algorithm", pp. 169–182 in *Advances in cryptology* (EUROCRYPT '84: Paris, 1984), edited by T. Beth et al., Lecture Notes in Comp. Sci. **209**, Springer-Verlag, New York, 1985.

[Pomerance et al. 1988] C. Pomerance, J. W. Smith, and R. Tuler, "A pipeline architecture for factoring large integers with the quadratic sieve algorithm", *SIAM J. Comput.* **17** (1988), 387–403.

[te Riele et al. 1989] H. J. J. te Riele, W. M. Lioen, and D. T. Winter, "Factoring with the quadratic sieve on large vector computers", *J. Comp. Appl. Math.* **27** (1989), 267–278.

[te Riele et al. 1991] H. te Riele, W. Lioen, and D. Winter, "Factorization beyond the googol with MPQS on a single computer", *CWI Quarterly* **4** (March 1991), 69–72.

[Riesel 1985] H. Riesel, *Prime numbers and computer methods for factorization*, Progress in Math. **126**, Birkhäuser, Boston, 1985.

[Silverman 1987] R. D. Silverman, "The multiple polynomial quadratic sieve", *Math. Comp.* **48** (1987), 329–339.

[Wagstaff 1993] S. S. Wagstaff, Jr., December 13, 1993. Regular letter to subscribers describing the state of the Cunningham factorization project.

Henk Boender, Department of Mathematics and Computer Science, University of Leiden, P.O. Box 9512, 2300 RA Leiden, The Netherlands (henkb@wi.leidenuniv.nl), and Centrum voor Wiskunde en Informatica, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands (henkb@cwi.nl)

Herman J. J. te Riele, Centrum voor Wiskunde en Informatica, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands (herman@cwi.nl)

Received October 10, 1995; accepted September 5, 1996