Factoring Integers Above 100 Digits Using Hypercube MPQS

F. Damm, F.-P. Heider, G. Wambach

University of Cologne

March 3, 1994

Abstract

In this paper we report on further progress with the factorisation of integers using the MPQS algorithm on hypercubes and a MIMD parallel computer with 1024 T805 processors. We were able to factorise a 101 digit number from the Cunningham list using only about 65 hours computing time. We give new details about the hypercube sieve initialisation procedure and describe the structure of the factor graph that saves a significant amount of computing time. At March 3rd, we finished the factorisation of a 104 digit composite.

1 Scope and Achievements

The integer factoring problem carries a long mathematical tradition. It is of cryptographic significance since about 16 years, when the first public-key cryptosystem algorithms appeared. Several of these rely on the difficulty of factoring and thus we need to have reliable data about factoring time when chosing parameter sizes for asymmetric cryptosystems and the many protocols using them.

One year ago, a Parsytec MIMD parallel computer with 1024 T805 microprocessors was installed at our university. At the same time, the second author gave a course on parallelisation of number theoretic algorithms which led us to our factorisation project. We chose the MPQS general purpose factoring algorithm and implemented a sieve initialisation procedure using hypercubes that had been suggested before but was not yet implemented. By the end of the last year, our work yielded the factorisation of a 97 digit cofactor of $12^{327}+1$ from the Cunningham list [BLSTW83] ($C97$ for short), the experiences of which we described before.

In the meantime we continued with the project and obtained the factorisation of the

101 digit cofactor of $5^{273} - 1$ ($C101$), namely

$$5\ 56028\ 11293\ 76381\ 05007\ 45276\ 70973\ 97005\ 19820\ 97448\ 09147$$
$$14501\ 62657\ 88991\ 51675\ 71596\ 62654\ 25103\ 75972\ 77807\ 81281 \quad ,$$

into the factors

$$3\ 76909\ 36915\ 50663\ 79919\ 58222\ 42521\ 83771$$

and

$$1\ 47523\ 02766\ 68658\ 47120\ 04868\ 63136$$
$$40646\ 58947\ 74138\ 14689\ 11613\ 43843\ 33811$$

of 36 and 66 digits. We could do this with about 65 hours processing time and won further insight into the parameter selection for MPQS. Just in time, we completed the factorisation of a composite cofactor of 104 digits ($C104$) of the Cunningham number $12^{291}+1$ into prime factors of 40 and 65 digits.

The size of the composite number to be factored determines an upper bound for the dimension of the hypercubes used. Moving from a 97 digit composite to a 104 digit composite enables us to increment the dimension by one which results in the expected reduction of the time needed for the preparation stage. For example, only 3 out of 128 processors were needed for the initialisation and the traversal of the hypercubes while the other processors did the sieve. Thus the initialization time took less than 2.5 % of the whole computing time, compared with 20–30 % in the classical MPQS.

Much of the computation time otherwise needed was saved by looking for cycles in the "factor graph", following suggestions in [LeMa90]. Since the paper of Lenstra and Manasse is very brief and no other work has been published, we describe the construction and structure of this graph in greater detail here.

The main findings are that all our cycles lie in 1 connected component of the graph and that most of the cyles are shorter than 10 edges. Furthermore, we found graphs containing a large star at the node corresponding to 1, followed by a second star at the node corresponding to the multiplier (if different from 1) and the rest of the edges scattered around very sparsely. An algorithm that repeatedly cuts off the leaves of the graph takes moderate computing time and produces a much smaller subgraph which is equivalent to the original one when searching and exploiting cycles.

2 Details and Results

2.1 The Hypercube Variation

We assume familiarity with the 'Multiple Polynomial Quadratic Sieve' (MPQS) algorithm ([Pome84],[Silv87]) and will sketch only the improved hypercube variation used by us.

Let N be the composite integer to be factored. After choosing a factorbase $\mathcal{F}$ of R primes $p_i, 1 \leq i \leq R$, and a sieve length M, a lot of quadratic polynomials $Q_{ab}(X) = a^2X^2 + 2bX + c$ are generated, $b^2 - N = a^2c$ and $|b| < \frac{a^2}{2}$. It follows that $Q_{ab}(X) \equiv (a^2X + b)^2a^{-2} \bmod N$. The requirement $|Q_{ab}(-M)| \approx |Q_{ab}(0)| \approx |Q_{ab}(M)|$ leads to the condition $a^2 \approx \sqrt{2N}/M$. For every such polynomial the roots modulo $p_i, 1 \leq i \leq R$, must be computed (the *preparation stage*), the interval $[-M, M[\cap\mathbb{Z}$ is sieved (the *sieve stage*), and the candidates are collected.

Now for every prime p in the factorbase let t_p be a square root of $N \bmod p$: $t_p^2 \equiv N \bmod p$. If p does not divide a, then

$$Q_{ab}(x) \equiv 0 \bmod p \Leftrightarrow x \equiv (-b \pm t_p)a^{-2} \bmod p.$$

The t_p's are independent of the Q_{ab} and will be computed only once, but $a^{-2} \bmod p$ for every a and for every $p \in \mathcal{F}$ has to be computed.

In [Silv87] the a's are (pseudo-)primes not divisible by any $p \in \mathcal{F}$. Montgomery (quoted in [PoST88]) and Peralta ([Pera92]) independently observed that if $a = \pi_1 \cdot \ldots \cdot \pi_l$ is the product of l primes π_i (such that N is a quadratic residue modulo π_i), there are 2^l different values $b \bmod a^2$ with $b^2 \equiv N \bmod a^2$. Since $Q_{ab}(x) = Q_{a(-b)}(-x)$, we get 2^{l-1} different polynomials with each a.

Given α_j, β_j with

$$\alpha_j^2 \equiv N \bmod \pi_j^2$$

and

$$\beta_j \equiv \begin{cases} 1 \bmod \pi_j^2 \\ 0 \bmod \pi_i^2 \quad , i \neq j \end{cases}$$

every b can be written uniquely as $b = \sum_{j=1}^{l} \delta_j\alpha_j\beta_j \bmod a^2$ where $\delta_j \in \{+1, -1\}$. We fix $\gamma_j = \pm\alpha_j\beta_j \bmod a^2$ such that γ_j is less than $\frac{a^2}{2}$.

R. Peralta further noticed that the solution set of $b^2 \equiv N \bmod a^2$ is structured like an l-dimensional hypercube $C_l = \mathbb{F}_2^l$, vertices corresponding to solutions b. Two vertices

are adjacent if the corresponding solutions b, b' differ at exactly one sign δ_j. He suggested to follow a certain hamiltonian cycle of $C_l = (\mathbb{F}_2^{l-1} \times \{0\}) \cup (\mathbb{F}_2^{l-1} \times \{1\})$ resulting in a sequence $k_i, 1 \leq i \leq 2^l - 1$, with $k_i = j$ if step i changes coordinate j, such that the tour

$$b_{i+1} = b_i + 2\mu_i\gamma_{k_i} \bmod a^2$$

with $\mu_i = +1$ or -1 depending on whether step i changes coordinate k_i from $-$ to $+$ or from $+$ to $-$ visits all vertices of the hypercube.

Whereas Peralta devised an algorithm requiring three additions and one multiplication modulo every prime p in the factorbase to step from b_i to b_{i+1}, we use precomputed tables of $2\gamma_j a^{-2} \bmod p$ and $(a^2 - 2\gamma_j)a^{-2} \bmod p$ for $1 \leq j \leq l$ and every p in the factorbase to obtain from the modular root $x_i = (-ba^{-2} \pm t_p a^{-2}) \bmod p$ of $Q_{ab_i}(X)$ the root

$$x_{i+1} = (x_i - 2\mu_i\gamma_{k_i}a^{-2}) \bmod p$$

of $Q_{ab_{i+1}}(X)$ in only one addition mod p. The cost for doing this results in the additional space consumption of $2lR$ integers. As a second improvement we sieve many hypercubes at a time.

Nearly factored candidates out of the sieve stage are relations of the form

$$q_1 q_2 \prod_{i=0}^{R} p_i^{c_i} \equiv z^2 \bmod N$$

with $q_i = 1$ or $q_i > p_R$ prime. If both $q_1 = q_2 = 1$ this is called a full relation, otherwise a partial relation. Exploiting the idea of using 'partial partial relations' [LeMa90] more than compensates for the negative effect that a-values from the polynomials used are divided by several primes in the factorbase. We will describe this part in more detail in the next but one section.

2.2 Sketch of the Parallelisation

The Parsytec GCel installed at our university's "Zentrum für Paralleles Rechnen" is a MIMD parallel computer and consists of 1024 Inmos T805 transputers. Every processor has only 4 MByte RAM, 350 kByte of which are occupied by the operating system. Therefore both parallel MPQS-implementations described in [CaSi88] and [Lens92] were not applicable on our machine.

We just mention the development of two parallel approaches particularly suited for MIMD parallel computers (a full description of both implementations including the communication aspects will be given in [Wamb94]). While the first one is presently used, the second one still is in an experimental phase. Both methods use a dedicated node (the 'root') whose only tasks consist in the collection of candidates and the input/output-operations.

In the first implementation we have two additional types of processors which we will call 'masters' and 'slaves'. Each master creates its own set of hypercubes. After initializing its first hypercube, the first master travels along the hamiltonian cycle described above. At every vertex it computes the new set of modular roots of Q_{ab}. These $2R$ integers and the coefficients of the polynomial are sent to a consecutive set of slaves which will sieve with the same polynomial. After making busy all slaves, the first master leaves its hypercube and initializes the second one. In the meantime the slaves that have finished are at the disposal of the second master, and so on. Each slave sieves its part of the sieve array with the received roots. Any candidates found after the sieving process will be stored in a local buffer. When the buffer overflows its content is sent to the root. Critical parameters here are the number of masters and the number of slaves in a consecutive set. These values obviously depend on the number to be factored (determining R and M), the dimension of the hypercubes and on the number of processors the program will run on. They are chosen carefully to avoid any idle times.

The second parallelisation groups all processors but the root together into rings of r processors each. Every group works on its own hypercubes. The factorbase is split into r parts of size R/r. Moving from one vertex to another, each processor in a group first computes its part of the modular roots of the new polynomial Q_{ab}. After $2(r-1)$ communications with its two neighbours involving $2R/r$ integers every processor knows the modular roots of Q_{ab} for the whole factorbase. The sieve array is split into r parts, too. Every processor sieves its part using blocks of predefined length. Any candidates found after the sieving process will be stored in a local buffer as in the first approach. While the amount of data sent among the processors is roughly the same, communicating processors are not as far apart as in the first approach, if the rings are mapped carefully on the processors obeying the underlying physical topology. Therefore it seems worthwile to let the root process runnning on the front-end computer and communicate to the root via sockets.

2.3 Structure in the Factor Graph of Composite Numbers

Let $\mathcal{F} = \{p_0, \ldots, p_R\}$ be the factorbase with $p_0 = -1$ and R primes used to factor a number N, let q, z, c_i be integers (c_i non-negative, $i = 0, \ldots, R$). We assume that no multiplier m is used and treat $m \neq 1$ separately in the next section.

number factorised	$C97$	$C101$	$C104$
sieve	9.437.184	12.582.912	15.728.640
factorbase	40.000	40.000	42.000
cubes	11.789	15.946	17.021
polynomials	671.913	712.524	2.110.604
computing time	39 h	65 h	177 h
full relations, initially	6.124	9.838	5.805
partial relations	1.207.688	1.328.204	3.337.419
pairs	7.055	11.155	5.780
full relations, including pairs	13.179	20.993	11.585

Table 1: Full Relations Found without Factor Graph

From the sieving stage we obtain a large number of partially factorised quadratic residues modulo N (we call them "partial relations", like [LeMa90])

$$q \prod_{i=0}^{R} {p_i}^{c_i} \equiv z^2 \bmod N \tag{1}$$

We know that all divisors of q are larger than p_R and we further have (assuming $p_R > 2^{16}$)

$$q \leq p_R^{\tau} \tag{2}$$

$$(p_R^2 < q < b^2 \text{ and } q \text{ is composite}) \quad \text{or} \quad q < 2^{32} < p_R^2 \tag{3}$$

τ , 2^{32} and the bound b are parameters of the algorithm. We used $b = 10^8$ with $C97$, $b = 2 \cdot 10^8$ with $C101$, $b = 3 \cdot 10^8$ with $C104$ and $\tau \leq 3$. The compositeness in (3) is tested with the Miller-Rabin algorithm.

If $q < p_R^2$, we immediately know that q must be a prime. We store the relation, if moreover $q < 2^{32}$. This is because we do not want to exceed one machine word. In a previous approach, we only stored partial relations with $q < p_R^2$, if $q < b$ held. The move

to $q < 2^{32}$ caused no significant extra cost and the number of cycles in the factor graph increased by about 10% (with $b = 10^8$, $C97$).

The second interesting case in (3) is $p_R^2 < q < b^2$ and q is not prime. Then it must consist of exactly two factors, say r and s:

$$q = r \cdot s, \qquad p_R < r, s < \frac{b^2}{p_R} \tag{4}$$

In a first step we store all the q-values of partial relations (1) in an array, sort it and look for double entries. Whenever a pair

$$\begin{aligned} q \prod_{i=0}^{R} {p_i}^{c_{1,i}} &\equiv z_1^2 \bmod N \\ q \prod_{i=0}^{R} {p_i}^{c_{2,i}} &\equiv z_2^2 \bmod N \end{aligned}$$

appears, we can easily obtain a full relation (to store) for the final MPQS step:

$$\prod_{i=0}^{R} {p_i}^{c_{1,i}+c_{2,i}} \equiv \left(\frac{z_1 z_2}{q}\right)^2 \bmod N$$

With $C97$, we obtained 7.055 pairs of this kind from 1.207.688 partial relations, while 1.328.204 partial relations produced 11.155 pairs with $C101$ and 3.337.419 partial relations made 5.780 pairs with $C104$. The computing time is additionally shown in table 1. The sizes of the factorbases were restricted by the program used for the last step of the MPQS algorithm and hence not optimal.

In the next step, we run a factorisation procedure on the q's in (4) to find r and s. We use the Pollard ρ-algorithm with at most $\lfloor\sqrt{b}\rfloor$ iteration steps using one polynomial. Whenever the algorithm finds a factor r, we calculate $s = q/r$ and check whether $r < 2^{32}$ and $s < 2^{32}$. If so, we store the relation for later use. Otherwise, we discard it.

With $C101$, we factored 886.334 q-values. 835.137 were found to factorise in two 32 bit parts, 49.442 contained a factor $\geq 2^{32}$ and 3.582 could not be factored in 14.142 steps.

Now we are prepared to introduce and build the factor graph $G = (V, E)$. The first node in V is identified with 1. All the primes q appearing in (1) and r, s appearing in (4, 1) make up the rest of the nodes. We put an edge $e = (v, w)$ into the graph, iff $v = 1$ and $w = q$, q a prime in (1) or $v = r$ and $w = s$ in (4). The graph of $C101$ contained 1.470.729 nodes and 1.264.025 edges.

number factorised	$C101$			
	nodes	edges	leaves	isolated nodes
full graph	1.470.729	1.264.025	1.197.630	—
1st optimisation	273.099	237.361	109.335	48.305
2nd	115.459		28.938	3.375
3rd	83.146		9.994	429
4th	72.723		3.842	61
5th	68.820		1.575	12
6th	67.233		617	4
7th	66.612		246	—
8th	66.366		100	—
9th	66.266		42	—
10th	66.224		23	—
11th	66.201		12	—
12th	66.189		6	—
13th	66.183		2	—
14th	66.181		1	—
15th	66.180	94.645	—	—

Table 2: Deletion of Superfluous Nodes from the Factor Graph

We basically use two graph algorithms to investigate and exploit the factor graph. The first one is a breadth-first search algorithm and the second one cuts off leaves and isolated nodes repeatedly. By breadth-first search we look for cycles in G, whenever a cycle is found, it is stored, the last edge traversed is deleted from G, and the search is continued.

We found that $G(C101)$ contained 235.170 connected components. The graph contains a large star at node 1 (about 431.000 edges) and a rest of about 832.000 edges distributed between 1.470.728 nodes. We found 28.466 cycles in the component that contains node 1. The deletion of all nodes that can not be part of cycles with repeated leaf-cutting is shown in table 2. (Components containing one edge only are counted as two leaves here.)

We can deduce that $G(C101)$ contained the reduced graph of 66.180 nodes and 94.645 edges in one single component, plus 1 tree of depth 14, 1 tree of depth 13, 4 trees of depth

12, ... , 63.937 trees of depth 2 and 1.197.630 leaves. Isolated nodes are left from tree components of G with even diameter, i.e. G contained 48.305 tree components with two edges diameter, 3.375 trees with 4 edges diameter, and so forth.

Using breadth-first search after fewer cutoff steps we found, that indeed most of the components only contain 1 edge. 25 cutoff steps are not prohibitive in computer time (about 15 minutes at an IBM RS 6000 / 53H) such that cycle search in even larger factor graphs should be feasible with moderate memory sizes.

By the above procedure we get a collection of cycles that are a basis of the cycle space of G. We do not find a basis of cycles whose lengths are shortest, which might be desirable because of the cycle use (see below). However, our cycle lengths do not seem to make that necessary: 9.868 were 3 edges long, 18.328 were between 4 and 9 edges and only 270 of them were from 10 to 20 edges long.

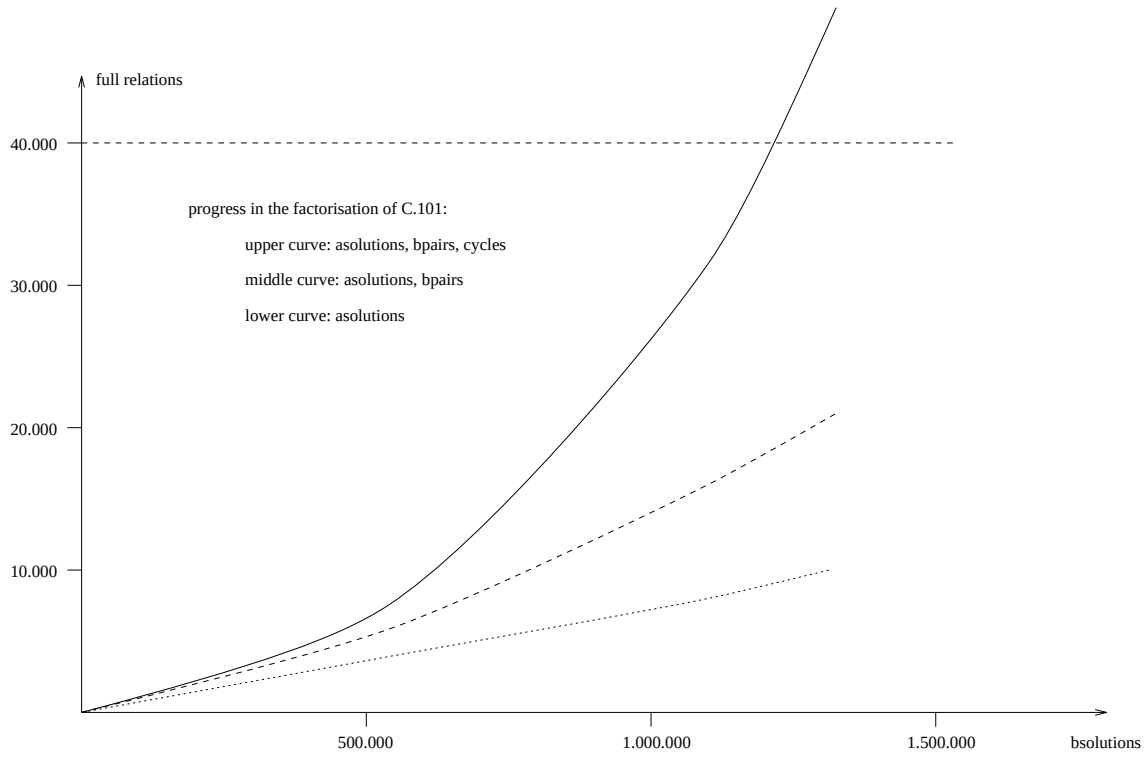

Figure 1: Development of the Overall Number of Full Relations with $C101$

Next, we can make a full relation out of each of the cycles found. Let us e.g. use a

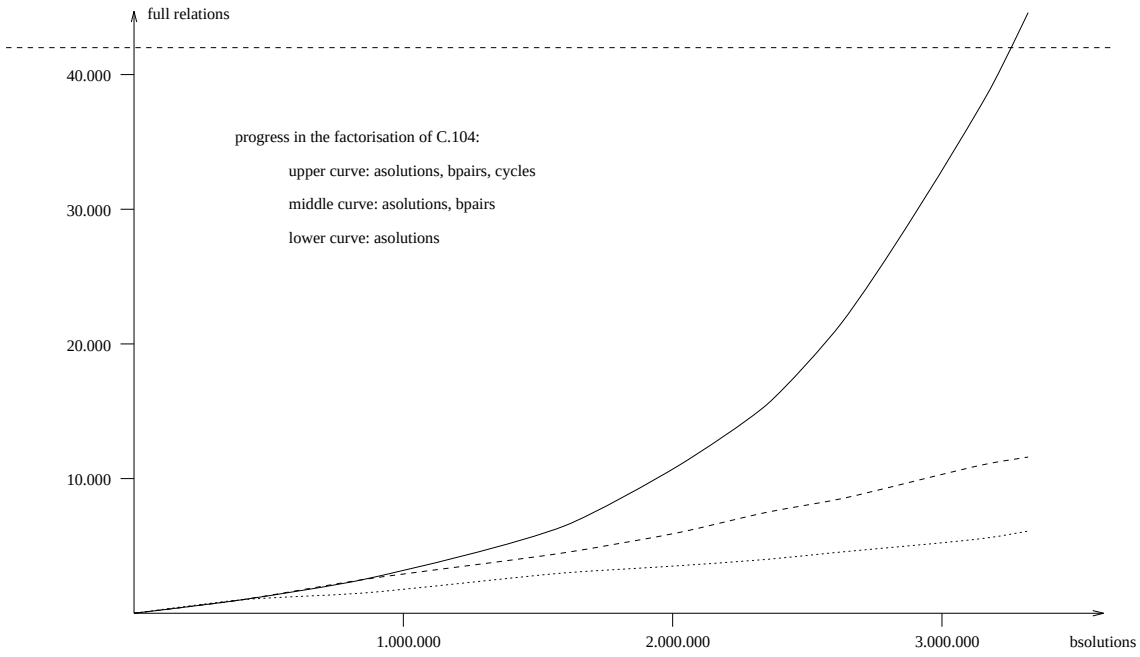

Figure 2: Development of the Overall Number of Full Relations with $C104$

cycle of length 4 passing node 1, that is a situation like

$$\begin{aligned} 1 \cdot r \cdot \prod_{i=0}^{R} {p_i}^{c_{1,i}} &\equiv z_1^2 \bmod N \quad , \quad r \cdot s \cdot \prod_{i=0}^{R} {p_i}^{c_{2,i}} \equiv z_2^2 \bmod N, \\ s \cdot t \cdot \prod_{i=0}^{R} {p_i}^{c_{3,i}} &\equiv z_3^2 \bmod N \quad , \quad 1 \cdot t \cdot \prod_{i=0}^{R} {p_i}^{c_{4,i}} \equiv z_4^2 \bmod N \end{aligned}$$

which gives us

$$\begin{aligned} z_1^2 \cdot z_2^2 \cdot z_3^2 \cdot z_4^2 &\equiv 1 \cdot r \cdot r \cdot s \cdot s \cdot t \cdot t \cdot 1 \cdot \prod_{i=0}^{R} {p_i}^{c_{1,i}+c_{2,i}+c_{3,i}+c_{4,i}} \bmod N \\ &= (r \cdot s \cdot t)^2 \cdot \prod_{i=0}^{R} {p_i}^{c_{1,i}+c_{2,i}+c_{3,i}+c_{4,i}} \bmod N \end{aligned}$$

and thus

$$\left(\frac{z_1 z_2 z_3 z_4}{rst}\right)^2 \equiv \prod_{i=0}^{R} {p_i}^{c_{1,i}+c_{2,i}+c_{3,i}+c_{4,i}} \bmod N$$

This way we obtained 28.466 (33.004) more full relations for $C101$ ($C104$), which was sufficient with 9.838 (5.805) and 11.155 (5.780) already available (table 3). The development of the overall number of full relations as partial relations were found is sketched in figures 1, 2.

number factorised	$C101$	$C104$
full relations, initially	9.838	5.805
partial relations	1.328.204	3.337.419
full relations, from pairs	11.155	5.780
factorised q's	1.318.525	3.333.786
prime	432.191	299.357
composite	835.137	2.453.363
factor too large	49.442	558.304
no factorisation	3.582	22.762
full graph:		
nodes	1.470.729	3.441.027
edges	1.264.025	2.750.587
components	235.170	723.444
reduced graph:		
nodes	66.180	100.130
edges	94.645	133.133
components	1	1
cycles (further full relations)	28.466	33.004
cycle length:		
3	9.868	5.769
$4 \ldots 9$	18.328	24.361
$10 \ldots 19$	270	2.865
$20 \ldots 49$		8
overall nr. of full relations	49.459	44.589

Table 3: Factor Graph and Full Relations of $C101$ and $C104$

2.4 Using a Multiplier $m \neq 1$

It is common to use a multiplier to speed up MPQS. To obtain more small primes in the factorbase, N is substituted by $m \cdot N$ for the construction of $\mathcal{F}$ and the sieve (with m a small integer). We (e.g.) used $m = 5$ to factorise $C97$ and report on our findings in this section.

Given that the size of the factorbase $\mathcal{F}$ is chosen first and the cycles in the factor graph are used, we did not experience the savings in computing time estimated by Pomerance et. al. [PoST88].

However, the factor graph gets an entirely different shape with $m \neq 1$. We found it containing two large stars at node 1 and node m now which are connected by a $(1, m)$-edge. Hence, short cycles like $(1, m)(m, q)(q, 1)$, with q a prime, are possible for many different q-values.

At first sight it seems to be a drawback that we have to treat 4 types of edges now instead of two: From partial relations like (1) we might obtain edges

$(1, q)$	with q a prime, $q < 2^{32}$,
(m, q)	with q a prime, $q < 2^{32}$,
(r, s)	with $q = r \cdot s$, r, s primes, both $< 2^{32}$,
$(r, s)_m$	with $q = m \cdot r \cdot s$, r, s primes, both $< 2^{32}$.

We introduced an m flag for every edge in G to recognize edges of the second and fourth type "containing" m. When factoring q from (1), we have to be careful whether possibly m^2 divides q. However, with $C97$ this did not occur.

The graph is exploited similar to the case $m = 1$. With every cycle we distinguish two cases. In the first case the cycle contains an even number of m-edges, e.g.

$$\begin{aligned} m \cdot r \cdot \textstyle\prod_{i=0}^{R} p_i^{c_{1,i}} &\equiv z_1^2 \bmod N \quad , \quad r \cdot s \cdot \textstyle\prod_{i=0}^{R} p_i^{c_{2,i}} \equiv z_2^2 \bmod N, \\ m \cdot s \cdot t \cdot \textstyle\prod_{i=0}^{R} p_i^{c_{3,i}} &\equiv z_3^2 \bmod N \quad , \quad 1 \cdot t \cdot \textstyle\prod_{i=0}^{R} p_i^{c_{4,i}} \equiv z_4^2 \bmod N \end{aligned}$$

Hence we get the full relation

$$\left(\frac{z_1 z_2 z_3 z_4}{mrst}\right)^2 \equiv \prod_{i=0}^{R} p_i^{c_{1,i}+c_{2,i}+c_{3,i}+c_{4,i}} \bmod N$$

The other case is a cycle containing an odd number of m-edges. We can not directly use the above method now, however we are allowed to combine two cycles with an odd

number of m-edges. More precisely, we store the "first odd m-cycle", say

$$\begin{aligned} m \cdot r \cdot \prod_{i=0}^{R} {p_i}^{c_{1,i}} &\equiv z_1^2 \bmod N \quad , \quad r \cdot s \cdot \prod_{i=0}^{R} {p_i}^{c_{2,i}} \equiv z_2^2 \bmod N, \\ s \cdot 1 \cdot \prod_{i=0}^{R} {p_i}^{c_{3,i}} &\equiv z_3^2 \bmod N \end{aligned}$$

and combine it to a full relation with every other odd m-cycle, say e.g.

$$\begin{aligned} m \cdot t \cdot u \cdot \prod_{i=0}^{R} {p_i}^{c_{4,i}} &\equiv z_4^2 \bmod N \quad , \quad m \cdot u \cdot v \cdot \prod_{i=0}^{R} {p_i}^{c_{5,i}} \equiv z_5^2 \bmod N, \\ m \cdot v \cdot \prod_{i=0}^{R} {p_i}^{c_{6,i}} &\equiv z_6^2 \bmod N \end{aligned}$$

yielding the full relation

$$\left(\frac{z_1 \ldots z_6}{m^2 rstuv}\right)^2 \equiv \prod_{i=0}^{R} {p_i}^{c_{1,i}+c_{2,i}+c_{3,i}+c_{4,i}} \bmod N$$

With $C97$, we found 36.489 full relations by this prodecure, which was far more than would have been needed.

2.5 Outlook

Due to space and time limitiations, this paper shows only the most interesting and important parts of our results. We continue our work on the second parallelisation [Wamb94] and the investigation of the structure of the factor graph [Damm94]. Of course, we are going to factor bigger numbers in the near future too. We would be very grateful for comments.

2.6 Acknowledgement

We are very grateful to R. Schrader for generous support. We owe further thanks to M. Behland for his help with the assembler programming, and to the "Zentrum für Paralleles Rechnen" for offering computing time.

References

[BLSTW83] J. Brillhart, D. H. Lehmer, J. L. Selfridge, B. Tuckerman, and S. S. Wagstaff, Jr., *Factorizations of $b^n \pm 1$ for b = 2, 3, 5, 6, 7, 10, 12, up to High Powers*. American Mathematical Society, Providence, Rhode Island, 1983.

[CaSi88] T. S. Caron, R. D. Silverman, "Parallel Implementation of the Quadratic Sieve", *Journal of Supercomputing*, 1 (1988), pp. 273-290.

[Damm94] F. Damm, "Cycle Structures in the Factor Graph of a Composite Number", *in preparation.*

[LeMa90] A. K. Lenstra, M. S. Manasse, "Factoring with two large primes" (Extended Abstract), *Advances in Cryptology, Eurocrypt '90*, Lecture Notes in Computer Science 473 (1991), pp.72-82.

[Lens92] A. K. Lenstra, "Massively Parallel Computing and Factoring", *Proceeding Latin '92*, Lecture Notes in Computer Science 583 (1992), pp.344 - 355.

[Pera92] R. Peralta, "A quadratic sieve on the n-dimensional cube", *Advances in Cryptology, Crypto '92*, Lecture Notes in Computer Science 740 (1993), pp.324-332.

[Pome84] C. Pomerance, "The Quadratic Sieve Factoring Algorithm", *Advances in Cryptology, Eurocrypt '84*, Lecture Notes in Computer Science 209 (1985), pp.169-182.

[PoST88] C. Pomerance, J. W. Smith, R. Tuler, "A pipeline architecture for factoring large integers with the quadratic sieve algorithm", *SIAM Journal of Computation*, Vol.17, No.2, pp.387-403, Apr. 1988.

[Silv87] R. D. Silverman, "The Multiple Polynomial Quadratic Sieve", *Mathematics of Computation*, Vol.48, No.177, pp.329-339, Jan. 1987.

[Wamb94] G. Wambach, "A Comparison of Two Parallelisations of the MPQS Algorithm on the Parsytec GCel", *in preparation.*