Complexity of Inverting the Euler Function

Scott Contini
Department of Computing
Macquarie University
Sydney, NSW 2109, Australia
contini@ics.mq.edu.au

Ernie Croot
School of Mathematics
Georgia Institute of Technology
Atlanta, GA 30332, USA
ecroot@math.gatech.edu

Igor E. Shparlinski
Department of Computing
Macquarie University
Sydney, NSW 2109, Australia
igor@ics.mq.edu.au

April 8, 2004

Abstract

We present an algorithm to invert the Euler function $\varphi(m)$. The algorithm, for a given integer $n \geq 1$, in polynomial time "on average", finds the set $\Psi(n)$ of all solutions m to the equation $\varphi(m) = n$. In fact, in the worst case the set $\Psi(n)$ is exponentially large and cannot be constructed by a polynomial time algorithm. In the opposite direction, we show, under some widely accepted number theoretic conjecture, that the problem of deciding whether $\varphi(m) = n$ for some m is **NP**-complete. Finally, we establish close links between the problem of inverting the Euler function and the integer factorisation problem.

1 Introduction

In this paper we study the complexity of a new number theoretic problem, namely the complexity of inverting the *Euler function* $\varphi(m)$, which, as usual, for an integer $m \geq 1$, is defined by

$$\varphi(m) = \#(\mathbb{Z}/m\mathbb{Z})^\times = \prod_{p^\alpha \| m} p^{\alpha-1}(p-1).$$

It is widely believed that computing the Euler function is equivalent to the integer factorisation problem. Moreover, let $\mathcal{P}_2$ denote the set of positive integers n which are products of two distinct primes p and q (with the additional condition $p \equiv q \equiv 3 \pmod 4$ such numbers are often called *Blum integers*). Then for $n \in \mathcal{P}_2$ finding $\varphi(n)$ is indeed equivalent to factoring n. Here we concentrate on the dual question of inverting the Euler function, which apparently has not yet been addressed in the literature. More precisely, given an integer $n \geq 1$, we want to find the set $\Psi(n)$ of all integer solutions $m \geq 1$ to the equation $\varphi(m) = n$.

Here we design an algorithm which solves this problem in exponential time in $\log n$ in the worst case, and in polynomial time for "almost all" n (provided the prime number factorisation of n is given). Because for infinitely many n the cardinality of $\Psi(n)$ is

exponentially large, any algorithm for inverting φ *must* run in time exponential in $\log n$ in the worst case (or nearly exponential). Indeed, from the proof of Theorem 4.6 of [14] we see that for infinitely many n,

$$\#\Psi(n) \ge n^{\gamma+o(1)}$$

where $\gamma > 0$ is any constant such that for any sufficiently large X there are at least $X^{1+o(1)}$ primes $p \le X$ such that all prime divisors of $p-1$ are less than $X^{1-\gamma}$ (see also [13]). By Theorem 1 of [3] one can take $\gamma = 0.7039$.

A natural question is whether the decision problem for inverting φ is any easier. We recall that n is called a *totient*, if there exists an integer m satisfying $\varphi(m) = n$. Given an integer n, and its prime factorization, how efficiently can we determine whether n is a totient? Because the output of any algorithm solving this problem need only be a single bit, we cannot so easily say that the running time must be exponential in $\log n$, as we did in the case of determining all the solutions m. We prove in Section 4 the somewhat surprising result that this decision problem is **NP**-complete, if we assume a certain strong form of the famous *Hardy–Littlewood* prime k-tuplet conjecture. Although at the present time this conjecture is out of reach, there are a number of results in this direction which leave little doubt that the conjecture is correct, for example, see [4].

Furthermore, in Section 5 we obtain an unconditional reduction from the problem of factoring integers $n \in \mathcal{P}_2$ to that of inverting the Euler function. As we have remarked, any polynomial time algorithm to compute the Euler function leads to a factorization algorithm for integers of the form $n = pq$ where p and q are primes. Here we prove a somewhat dual statement by showing that any polynomial time algorithm to invert the Euler function (in the sense that the running time is $(\#\Psi(n) + \tau(n) + \log n)^{O(1)}$), where the factorization of n is *not* given, leads to a probabilistic polynomial time factorisation algorithm for $n \in \mathcal{P}_2$. This result is certainly weaker than that of Section 4 but is not based on any unproven assumptions.

The growth, distribution in arithmetic progressions and in other special sets of elements of the values of the Euler function, and many other similar questions, have extensively been studied in the literature, see [5, 6, 9, 10, 11, 12, 13, 14] and references therein. Nevertheless the considered here questions seem to be new and have never been studied. We also remark that analogues of our results can be obtained for the sum of divisors function $\sigma(m)$ and for several more similar number theoretic functions.

2 Notation

We use $\omega(m)$ and $\tau(m)$ to denote the total number of distinct prime and positive integer divisors of a positive integer m, respectively (we also define $\omega(1) = 0$, $\tau(1) = 1$).

We also use the Vinogradov symbols $\gg$, $\ll$, $\asymp$ as well as the Landau symbols O and o with their regular meanings (we recall that $U \ll V$ and $U = O(V)$ are both equivalent to the inequality $|U| \le cV$ with some constant $c > 0$ and $U \asymp V$ is equivalent $U \ll V \ll U$). The implied constants in the symbols O, $\gg$, $\ll$ and $\asymp$ are always absolute unless indicated otherwise.

3 Constructing $\Psi(n)$

Our algorithm to find $\Psi(n)$ makes use of the prime power factorization of n. If we were to modify our algorithm to find $\Psi(n)$ where the factorisation of n is not given, but is first

found by using a probabilistic factoring algorithm (see [8]), then for most integers n, factoring would dominate the overall complexity of the algorithm. In the worst case, however, where $\Psi(n)$ is "large", the running time of the rest of the algorithm would dominate this factoring step. For our algorithm, we simply assume that the prime number factorisation of n is given, which has the additional advantage of making our algorithm deterministic (while making factoring n a part of the algorithm would make it probabilistic).

Theorem 1. *There exists a deterministic algorithm which given the prime number factorisation*

$$n = p_1^{\alpha_1} \dots p_s^{\alpha_s}$$

of an integer $n \geq 2$, constructs $\Psi(n)$ in time

$$T(n) \leq (\#\Psi^*(n) + \tau(n) + \log n)^{O(1)},$$

where

$$\Psi^*(n) = \bigcup_{d|n} \Psi(d).$$

Proof. Basically, we give an algorithm which efficiently finds all representations of n of the following type:

$$n = \prod_{j=1}^{k} \ell_j^{\gamma_j}(\ell_j - 1),$$

where $\ell_1 < \dots < \ell_k$ are primes, and where $\gamma_1, \dots, \gamma_k \geq 0$ are integers. Each such representation corresponds to a solution $\varphi(m) = n$, where

$$m = \prod_{j=1}^{k} \ell_j^{\gamma_j+1}.$$

Our algorithm is iterative and builds a graph, where all the vertices on the jth level correspond to a certain list $\mathcal{E}_j$, and where each solution m to $\varphi(m) = n$ corresponds to some path from a vertex back to the top list $\mathcal{E}_1$, although not all such paths correspond to such a solution. The vertices in each of these lists are assigned a certain value, which is an ordered pair of the form (ℓ_j, γ_j), where ℓ_j is prime, and where $\gamma_j \geq 0$ is an integer.

Given an integer n, we let $\mathcal{D}(n)$ denote the set of divisors of n. If we are given the prime power factorisation of n, then we can easily construct the set $\mathcal{D}(n)$ in time $\tau(n)^{O(1)}$.

We now describe $\mathcal{E}_1$: We let $\mathcal{E}_1$ be a set of vertices, one for each ordered pair (ℓ_1, γ_1), where ℓ_1 is a prime, and $\gamma_1 \geq 0$, such that the number $e = \ell_1^{\gamma_1}(\ell_1 - 1)$ lies in $\mathcal{D}(n)$; that is, $e|n$. We also remark that for every e there are at most two possible pairs (ℓ_1, γ_1). The vertices in this list are not be linked to each other, but are each doubly linked to entries of the yet to be mentioned list $\mathcal{E}_2$.

The list $\mathcal{E}_2$ is created as follows: We scan through $\mathcal{E}_1$, and for each vertex v in $\mathcal{E}_1$, having the value (ℓ_1, γ_1), we consider the integer $n_0 = n/\ell_1^{\gamma_1}(\ell_1 - 1)$. Then, among the integers $d_0 \in \mathcal{D}(n_0)$ (divisors of n_0), we locate all those corresponding to vertices $v \in \mathcal{E}_1$ having value (ℓ, γ), with $\ell > \ell_1$; that is, $d_0 = \ell^\gamma(\ell - 1)$. In this way, we run through all the divisors d of n of the form

$$d = \ell_1^{\gamma_1}(\ell_1 - 1)\ell^\gamma(\ell - 1), \quad \ell_1 < \ell \text{ are prime.}$$

The list $\mathcal{E}_2$ then consists of one vertex for each of these different ordered pairs (ℓ, γ), for each of the vertices $v \in \mathcal{E}_1$; and, this vertex is doubly linked to its ancestor $v \in \mathcal{E}_1$.

We note that each vertex in $\mathcal{E}_2$ has a unique ancestor; and, different vertices in $\mathcal{E}_2$ may have the same value (ℓ_2, γ_2).

In general, suppose we have constructed the list $\mathcal{E}_j$. Then, the list $\mathcal{E}_{j+1}$ is constructed as follows: By running through the vertices $v \in \mathcal{E}_j$, and then considering the unique path from v back to its ancestors in $\mathcal{E}_{j-1}, \mathcal{E}_{j-2}, \dots, \mathcal{E}_1$, we get that these vertices (along with v) correspond to a sequence of ordered pairs $(\ell_j, \gamma_j), \dots ., (\ell_1, \gamma_1)$, which represents a divisor d of n of the form

$$d = \prod_{i=1}^{j} \ell_i^{\gamma_i}(\ell_i - 1), \ \ell_1 < \ell_2 < \cdots < \ell_j \text{ are prime.}$$

We let $n_0 = n/d$, and then we scan through the set $\mathcal{D}(n_0)$, looking for elements of the form $\ell^\gamma(\ell - 1)$, where $\ell > \ell_j$ is a prime number. We then add a vertex to $\mathcal{E}_{j+1}$, assign it the value (ℓ, γ), and doubly link it to the vertex $v \in \mathcal{E}_j$. After we have done this for all these ordered pairs (ℓ, γ) generated by considering all $v \in \mathcal{E}_j$, the construction of $\mathcal{E}_{j+1}$ is completed.

We continue constructing these lists, until we reach a list $\mathcal{E}_t$ having no children. Since n has $O(\log n)$ prime power factors, and since each new level in the graph corresponds to a string of divisors $d_1, \dots, d_t$ where $d_1 \cdots d_t | n$, we conclude that $t = O(\log n)$.

It is obvious that each path from a vertex back to $\mathcal{E}_1$ along its unique ancestors in the graph corresponds either to a proper divisor

$$d = \prod_{j=1}^{h} \ell_j^{\gamma_j}(\ell_j - 1), \ \ell_1 < \ell_2 < \cdots < \ell_h \text{ are prime}$$

of n such that there is no pair (ℓ, γ), $\ell > \ell_h$ prime, $\gamma \geq 0$, with $\ell^\gamma(\ell - 1)d | n$; or, we have that $d = n$. Now, if $d = n$, then d corresponds to the solution

$$m = \prod_{j=1}^{h} \ell_j^{\gamma_j + 1}$$

of $\varphi(m) = n$. Thus, by considering the paths from vertices corresponding to n back to $\mathcal{E}_1$ corresponding to $d = n$, we obtain the set $\Psi(n)$.

Finally, it is obvious that the running time of the algorithm is proportional to

$$(L + \tau(n) + \log n)^{O(1)},$$

where L is the number of paths throughout the above graph which is $\#\Psi^*(n)$. □

To address the average performance of the algorithm, we require the following bound:

Theorem 2. *The bound following bound holds:*

$$\sum_{n \leq x} \#\Psi^*(n) \ll x \log x.$$

Proof. We have that

$$\sum_{n\le x} \#\Psi^*(n) = \sum_{n\le x}\sum_{d\mid n} \#\Psi(d) \le x \sum_{d\le x} \frac{\#\Psi(d)}{d}. \tag{1}$$

Now,

$$\sum_{d\le x} \#\Psi(d) = \#\{n \ge 1 \ : \ \varphi(n) \le x\} = \left(\frac{\zeta(2)\zeta(3)}{\zeta(6)} + o(1)\right) x,$$

see [7]. So, by partial summation, we conclude that

$$\sum_{d\le x} \frac{\#\Psi(d)}{d} = O(\log x),$$

which, together with (1) finishes the proof. □

An almost immediate corollary of Theorem 2, together with the well known bound

$$\sum_{n\le x} \tau(n) = O(x \log x), \tag{2}$$

see Theorem 2 in Section I.3.2 of [15], and Theorem 1, is the following:

Corollary 3. *For every $A > 0$, there exists $B > 0$, so that for all but at most $O(x/\log^A x)$ integers $n \le x$ we have that the algorithm in Theorem 1 finds $\Psi(n)$ in time $\log^B n$.*

4 NP-completeness of Totient Testing

A natural question is whether it is any easier to decide if, given n, there exists an integer m satisfying $\varphi(m) = n$. In this section we prove that this problem is **NP**-complete, if we assume the following strong form of the *Hardy–Littlewood* prime k-tuplet conjecture, see [4] for several results in this direction.

Conjecture 4. *There exists an integer $A > 0$ such that the following holds: Suppose that $(M_1x+a_1)(M_2x+a_2)$ has no fixed prime divisors as x runs through the integers, and that $M_1, M_2 > 0$, and $0 \le a_i < M_i$ for $i = 1$ and 2. Then, there exists an $x < \log^A(M_1M_2+1)$ such that both $M_1x + a_1$ and $M_2x + a_2$ are prime.*

We first note that the decision problem is in **NP**, since if we let $\mathcal{L}$ be the language consisting of all integers $n \ge 1$ such that there exists m satisfying $\varphi(m) = n$, then for each $n \in \mathcal{L}$ there exists a string s, of length $\log^{O(1)} n$, which we can use to verify that $n \in \mathcal{L}$ in polynomial time; in particular, such a string s is the prime power factorization of any solution m (and, given the prime power factorization of m, it is easy to compute $\varphi(m)$). Since we can check whether a number is prime in polynomial time, and therefore check that s is a legitimate prime power factorization in time $\log^{O(1)} m$, we conclude that $\mathcal{L}$ is in **NP**.

The problem which we reduce to our decision problem is the following variant of the subset sum problem, which is known to be **NP**-complete.

> Partition Problem: Given $2k \ge 2$ nonnegative integers $x_1, \dots, x_{2k}$, where $S = x_1 + \cdots + x_{2k}$ is even, decide whether there exist $1 \le i_1 < \ldots < i_k$ with $x_{i_1} + \cdots + x_{i_k} = S/2$.

Assuming Conjecture 4, we show there is a polynomial time reduction of the PARTITION problem to the problem of deciding whether there exist integers m satisfying $\varphi(m) = n$, for a certain small set of values of n.

To prove this theorem, we require the following result which could be of independent interest.

Theorem 5. *Given an odd number $k \geq 1$ and given $2k$ integers $x_1, \dots, x_{2k}$, we can construct in polynomial time a series of congruence classes $a_i \pmod M$, $(a_i, M) = 1$, such that if $N_1, \dots, N_{2k}$ are any numbers satisfying $N_i \equiv a_i \pmod M$, and if $\{i_1, \dots, i_\ell\} \subset \{1, \dots, 2k\}$, with $\ell \leq k$, then*

$$\gcd(2N_{i_1} \cdots N_{i_\ell} + 1,\ M) = 1 \quad \Longleftrightarrow \quad \ell = k \text{ and } x_{i_1} + \cdots + x_{i_\ell} = S/2; \tag{3}$$

$$N_i - 1 \nmid 4N_1 \cdots N_{2k}, \qquad i = 1, \dots, 2k; \tag{4}$$

$$\gcd(2N_1 \cdots N_{2k} + 1,\ M) > 1, \quad \text{and} \quad \gcd(4N_1 \cdots N_{2k} + 1,\ M) > 1. \tag{5}$$

Proof. First, we let $R_1, \dots, R_{k-1}$ be the first consecutive primes greater than k. Next, given

$$A = 1 + \sum_{i=1}^{2k} |x_i|,$$

we let $U_1, \dots, U_t$ be the first consecutive primes greater than R_{k-1} such that

$$\prod_{i=1}^{t} U_i > 2A.$$

Finally, we let $v = U_t$, and then let $V_1, \dots, V_v$ be consecutive primes greater than U_t. Then, we let

$$M = 8 \cdot 3 \cdot 5 \cdot \prod_{h=1}^{k-1} \frac{2^{R_h} + 1}{3} \prod_{i=1}^{t} \prod_{j=1}^{v} \frac{2^{U_i V_j} - 1}{(2^{U_i} - 1)(2^{V_j} - 1)}.$$

We claim that this integer M satisfies $\log M = (Ak)^{O(1)}$, which can be proved by repeated use of the Prime Number Theorem; also, we claim that each of these factors are coprime to the others, which can be proved by repeated use of the fact that $(2^H - 1, 2^K - 1) = 2^{(H,K)} - 1$.

We let $a_1, \dots, a_{2k}$ all be in the same class modulo $8 \cdot 3 \cdot 5$, defined via the Chinese remainder theorem as follows:

$$a_i \equiv 1 \pmod 8; \qquad a_i \equiv 2 \pmod 3; \qquad a_i \equiv 4 \pmod 5;$$

and, for $j = 1, \dots, k-1$, we let

$$a_i \equiv 2^{g_j} \pmod{(2^{R_j} + 1)/3}, \tag{6}$$

where for g_j is any solution to $1 + jg_j \equiv R_j \pmod{2R_j}$ (for j odd there is a unique g_j; and for j even, there are two values g_j that satisfy this).

The congruence condition modulo 8 ensures that (4) holds; the congruence modulo 3 forces the first part of (5) to hold; and the condition modulo 5 forces the second part

of (5) to hold. Finally, the condition (6) ensures that $\gcd(2N_{i_1}\cdots N_{i_\ell}+1, M) = 1$ implies $\ell = k$, which is part of (3).

Now, for $i = 1, 2, \ldots, t$, we let

$$\{\theta(i,1), \ldots, \theta(i, U_i - 1)\} = \{0, \ldots, U_i - 1\} \setminus \{S/2 \pmod{U_i}\};$$

that is, for every $i = 1, 2, \ldots, t$, the values of $\theta(i,j)$ run through the congruence classes modulo U_i, omitting the class $S/2 \pmod{U_i}$. Next, let

$$\delta_{i,j} \equiv k^{-1} \pmod{U_iV_j}, \quad 0 \le \delta_{i,j} \le U_iV_j - 1.$$

Then, for $i = 1, 2, \ldots, t$, $j = 1, 2, \ldots, U_i - 1$, and $\ell = 1, \ldots, 2k$, we let

$$a_\ell \equiv -2^{V_j x_\ell + \delta_{i,j}(V_j\theta(i,j)-1)} \pmod{\frac{2^{U_iV_j}-1}{(2^{U_i}-1)(2^{V_j}-1)}}$$

Then, if $\{x_{n_1}, \ldots, x_{n_k}\}$ is any k-element subset of k of $\{x_1, \ldots, x_{2k}\}$ such that $x_{n_1} + \cdots + x_{n_k} \ne S/2$, we must have that for some $i = 1, 2, \ldots, t$ and $j = 1, 2, .., U_i - 1$,

$$x_{n_1} + \cdots + x_{n_k} \equiv \theta(i,j) \pmod{U_i};$$

and so, on letting $T = (2^{U_iV_j} - 1)/(2^{U_i} - 1)(2^{V_j} - 1)$, we see that if $N_i \equiv a_i \pmod{M}$, then

$$\begin{aligned} 2N_{n_1}\cdots N_{n_k} + 1 &\equiv (-1)^k 2^{1+V_j(x_{n_1}+\cdots+x_{n_k}-k\delta(i,j)\theta(i,j))-k\delta(i,j)} + 1 \\ &\equiv -2^{U_iV_jI} + 1 \equiv 0 \pmod{T}, \end{aligned}$$

where I is some integer. Conversely, if $x_{h_1} + \cdots + x_{h_k} = S/2$, then one can show that $(2N_{h_1}\cdots N_{h_k} + 1, M) = 1$. Thus, we have established (3), and the result follows. □

Now are now ready to prove our main result.

Theorem 6. *Suppose that $x_1, \ldots, x_{2k}$ is an input of the* Partition *problem. Let*

$$B = \sum_{i=1}^{2k} \log(x_i + 2).$$

Then, in polynomial time, we construct a set of $s = B^{O(1)}$ integers $n_1, \ldots, n_s$ such that the answer to the corresponding Partition *problem is "Yes" if and only if for some $i = 1, 2, \ldots, s$ we have that $\varphi(m) = n_i$ has a solution.*

Proof. Suppose $x_1, \ldots, x_{2k}$ are given. We may assume that k is odd, since if k is even, then we can enlarge our set $\{x_1, \ldots, x_{2k}\}$ by two new elements $x_{2k+1} = x_{2k+2} = 0$.

Now, suppose that $p_1, \ldots, p_{2k}$ are a set of primes satisfying $p_i \equiv a_i \pmod{M}$, $p_i > M$. Then, as a consequence of (3), (4), and (5) of Theorem 5, one can see that if there is a solution m to

$$\varphi(m) = 4p_1\cdots p_{2k},$$

then $m = P_1P_2$ or $2P_1P_2$, where P_1 and P_2 are both primes satisfying

$$P_1 = 2p_{i_1}\cdots p_{i_k} + 1, \quad \text{and} \quad P_2 = 2p_{j_1}\cdots p_{j_k} + 1,$$

where $\{p_{i_1}, \ldots, p_{i_k}\} \cup \{p_{j_1}, \ldots, p_{j_k}\} = \{p_1, \ldots, p_{2k}\}$. Moreover, we have

$$x_{i_1} + \cdots + x_{i_k} = S/2 = x_{j_1} + \cdots + x_{j_k}. \tag{7}$$

Now suppose that there are two subsets of $\{x_1, \ldots, x_{2k}\}$ satisfying (7). Let ℓ be one of the numbers $2, 3, \ldots, k+2$, and suppose we are lucky and have $1 \in \{i_1, \ldots, i_k\}$ and $\ell \in \{j_1, \ldots, j_k\}$, or have $1 \in \{j_1, \ldots, j_k\}$ and $\ell \in \{i_1, \ldots, i_k\}$; certainly, for one of these values $\ell = 2, 3, \ldots, k+2$ this must hold. We suppose that $1 \in \{i_1, \ldots, i_k\}$ and $\ell \in \{j_1, \ldots, j_k\}$. Let $\{t_1, \ldots, t_{2k-2}\} = \{1, 2, \ldots, 2k\} \setminus \{1, \ell\}$. Then, assuming conjecture 4 (speicializing to the case of one linear form, instead of two), we can pick values $t_1, \ldots, t_{2k-2} < B^{O(1)}$ such that the numbers $a_i + Mt_i$ are all prime; moreover, we can pick these numbers in time $B^{O(1)}$, by first picking t_1, then t_2, and so on.

Now, we consider the polynomials

$$F(x) = 2(a_1 + Mx) \prod_{\substack{u \in \{i_1, \ldots, i_k\} \\ u \neq 1}} (a_u + Mt_u) + 1,$$

and

$$G(y) = 2(a_\ell + My) \prod_{\substack{u \in \{j_1, \ldots, j_k\} \\ u \neq \ell}} (a_u + Mt_u) + 1.$$

By (3), $F(x)$ and $G(y)$ are coprime to M for all integers x, y, and so have no fixed prime divisors; moreover, $(a_1 + Mx)F(x)$ and $(a_\ell + My)G(y)$ have no fixed prime divisors. So, assuming Conjecture 4, if we run through the values $x, y < B^{O(1)}$ that make $a_1 + Mx$ and $a_\ell + My$ both prime, then among these values x and y, there must be a choice which makes $a_1 + Mx, a_\ell + My, F(x)$, and $G(y)$ all prime. So, we have a set of primes $p_1, \ldots, p_{2k}$ of the form

$$p_1 = a_1 + Mx, \qquad p_\ell = a_\ell + My,$$

and

$$p_i = \ a_i + Mt_i, \qquad i = 2, \ldots, \ell - 1, \ell + 1, \ldots, 2k.$$

These primes satisfy the congruence conditions $p_i \equiv a_i \pmod{M}$. Furthermore, we also have that $2p_{i_1} \cdots p_{i_k} + 1 = F(x)$ is prime, as is $2p_{j_1} \cdots p_{j_k} + 1 = G(y)$. So, if we let $n(x, y) = 4p_1 \cdots p_{2k}$, then we get a solution $\varphi(F(x)G(y)) = n(x, y)$. So, by running through choices for $x, y < B^{O(1)}$, and $\ell = 2, 3, \ldots, k+2$, we are guaranteed to hit upon a value $n(x, y)$ having a solution $\varphi(m) = n(x, y)$, as long as there is a subset of $\{x_1, \ldots, x_{2k}\}$ summing to $S/2$.

Conversely, if there is no subset of $\{x_1, \ldots, x_{2k}\}$ summing to $S/2$, then either $F(x)$ or $G(y)$ is an odd composite number, and so they fail to satisfy $\varphi(F(x)G(y)) = n(x, y)$ for all values x, y.

Thus, the PARTITION problem can be reduced, in polynomial time, to the problem of deciding whether $\varphi(m) = n$ for a set of $B^{O(1)}$ values n, which finishes the proof. □

5 Inverting the Euler Function and Integer Factorisation

The algorithm of Theorem 1 assumes that the prime number factorisation of n is given. Here we show the factorisation problem for integers from $\mathcal{P}_2$ can be reduced in in probabilistic polynomial time to the problem of inverting the Euler function.

Theorem 7. *Given an algorithm that finds $\Psi(m)$ in time $(\#\Psi^*(m)+\tau(m)+\log m)^{O(1)}$, without being given the prime factorisation of n, one can factor integers $n \in \mathcal{P}_2$ in probabilistic polynomial time.*

Proof. Let $\pi(X; r, a)$ denote the number of primes $\ell \le X$ with $\ell \equiv a \pmod r$. We need the following result which is a greatly relaxed version of Theorem 2.1 of [2]. Namely, if r is a sufficiently large prime number then for $X \ge r^3$

$$\pi(X, 4r, a) \ge \frac{X}{4r \log X}. \tag{8}$$

for any integer a with $\gcd(a, 4r) = 1$.

Now, assume we are given sufficiently large odd $n = pq \in \mathcal{P}_2$. We choose two positive integers $k_1, k_2 \le n^3$ and consider the product $4(2k_1 + 1)(2k_2 + 1)n$.

It is clear that if $4(2k_1 + 1)(2k_2 + 1)n = \varphi(m)$ then $\omega(m) \le 3$. More precisely, it is possible only for the values of m of the form

1. $m = \ell$ or $m = 2\ell$ or $m = 4\ell$ where ℓ is prime;

2. $m = \ell_1\ell_2$ or $m = 2\ell_1\ell_2$ where ℓ_1, ℓ_2 are prime;

In each case of the first group ℓ is uniquely defined (and clearly there are at most two suitable values of ℓ).

Both cases of the second type occur simultaneously with the same values of ℓ_1, ℓ_2 which (up to a permutation) are either of the form

$$\ell_1 = 2d_1 + 1, \qquad \ell_2 = 2d_2pq + 1,$$

or of the form

$$\ell_1 = 2d_1p + 1, \qquad \ell_2 = 2d_2q + 1,$$

where d_1 and d_2 are divisors of $(2k_1+1)(2k_2+1)$ with $d_1d_2 = (2k_1+1)(2k_2+1)$. Therefore, there are at most

$$2\tau((2k_1 + 1)(2k_2 + 1)) \le 2\tau(2k_1 + 1)\tau(2k_2 + 1)$$

possible solutions of the second kind. We see from (2) then the total number of positive integers $k \le X$ with $\tau(k) \ge \log^3 X$ is $O(X \log^{-2} X)$. Thus from (8) (applied with $r = p$ and $a = 2r + 1$) we derive that there are at least

$$\frac{4n^3p}{4p \log n^3} + O(n^3 \log^{-2} n) \ge \frac{n^3}{2 \log n^3}$$

positive integers $k_1 \le n^3$ for which simultaneously $2(2k_1+1)p+1$ is prime and $\tau(2k_1+1) \le \log^3 n$. Similarly, we have at least the same number of positive integers $k_2 \le n^3$ for which simultaneously $2(k_2 + 1)q + 1$ is prime and $\tau(2k_2 + 1) \le \log^3 n$.

For each such pair of integers k_1, k_2 we see that the cardinality of $\Psi(4(2k_1+1)(2k_2+1)n)$ is polynomially bounded, namely, $\#\Psi(4(2k_1 + 1)(2k_2 + 1)n) = O(\log^6 n)$, and contains a solution of the form

$$m = (2(2k_1 + 1)p + 1)(2(2k_2 + 1)q + 1) \tag{9}$$

from which, together with the equation $n = pq$, the primes p and q can be trivially found (we certainly have to try all values of $m \in \Psi(4(2k_1 + 1)(2k_2 + 1)n)$ in order to find the one of the form (9)).

These considerations naturally lead to the following probabilistic algorithm which finds the above pair of k_1, k_2 and thus the primes p and q.

Assume that the inverting algorithm outputs $\Psi(N)$ in time bounded by $(\#\Psi(N) \log N)^A$ for some constant $A > 0$. We choose integers k_1, k_2 uniformly at random in the interval $[1, n^3]$ and use the algorithm to compute $\Psi(4(2k_1 + 1)(2k_2 + 1)n)$. If the time it takes exceeds $\log^{8A} N$ this means that $\#\Psi(4(2k_1 + 1)(2k_2 + 1)n) \geq \log^7 N$ and we simply terminate the algorithm and choose another pair k_1, k_2. It is clear that in the expected time $O(\log^6 n)$ we find the desired pair of k_1, k_2. □

References

[1] M. Agrawal, N. Kayal and N. Saxena, 'PRIMES is in **P**', *Preprint*, 2002, 1–9.

[2] W. R. Alford, A. Granville and C. Pomerance, 'There are infinitely many Carmichael numbers', *Annals of Math.*, **140** (1994), 703–722.

[3] R. C. Baker and G. Harman, 'Shifted primes without large prime factors', *Acta Arith.*, **83** (1998), 331–361.

[4] A. Balog, 'The prime k-tuplets conjecture on average', *Analytic Number Theory*, Progress in Mathematics **85**, Birkhäuser, Boston, 1990, 47–75.

[5] W. Banks, J. B. Friedlander, C. Pomerance and I. E. Shparlinski, 'Multiplicative structure of values of the Euler function', *Proc. Conf. in Number Theory in Honour of Prof. H.C. Williams*, 2003, (to appear).

[6] W. Banks, F. Luca, F. Pappalardi and I. E. Shparlinski, 'Values of the Euler function in various sequences', *Preprint*, 2004.

[7] P. T. Bateman, 'On the distrribution of values of the Euler function', *Acta Arith.*, **21** (1972), 329–345.

[8] R. Crandall and C. Pomerance, *Prime numbers: A Computational perspective*, Springer-Verlag, Berlin, 2001.

[9] T. Dence and C. Pomerance, 'Euler's function in residue classes', *The Ramanujan J.*, **2** (1998), 7–20.

[10] P. Erdős and C. Pomerance, 'On the normal number of prime factors of $\varphi(n)$', *Rocky Mountain J. Math.*, **15** (1985), 343–352.

[11] K. Ford, 'The number of solutions of $\varphi(x) = m$', *Annals of Math.*, **150** (1999), 283–311.

[12] K. Ford, S. Konyagin and C. Pomerance, 'Residue classes free of values of Euler's function', *Proc. Number Theory in Progress*, Walter de Gruyter, Berlin, 1999, 805–812.

[13] C. Pomerance, 'Popular values of Euler's function', *Mathematika*, **27** (1980), 84–89.

[14] C. Pomerance, 'Two methods in elementary analytic number theory', *Number theory and application*, R. A. Mollin, ed., Kluwer Acad. Publ., Dordrecht, 1989, 135–161.

[15] G. Tenenbaum, *Introduction to analytic and probabilistic number theory*, Cambridge University Press, UK, 1995.