Comparing characterizations of Carmichael numbers for computation

Simon Johansson

Abstract. This paper examines the properties of Carmichael numbers with the aim of constructing an algorithm for computation. This includes a recently published characterization, whose applications in computing are studied. The resulting algorithm is then described and a run time comparison with an older algorithm is presented.

1. Introduction

With the practical application of prime numbers in modern cryptography, the search for primes has gained a lot of attention outside of mathematics. Many algorithms have been developed to test primality as efficiently as possible and in 2002 Agrawal et al. managed to develop the AKS primality test, which is a deterministic test able to check primality in polynomial time [1]. The history of prime testing stretches far back, however, and Pierre de Fermat famously stated in 1640 [2]:

Theorem 1. *(Fermat's little theorem) If p is a prime, then for any integer b coprime to p*

$$b^{p-1} \equiv 1 \ (mod\, p) \tag{1}$$

This begs the question: Is this a property unique for prime numbers? In other words, could we with certainty determine the primality of a number by checking if the congruence is satisfied for all integers b? The answer proves to be no, as there is a set of composite numbers for which Fermat's little theorem also holds. These numbers, the "Carmichael numbers", are a challenge in the construction of primality tests and more knowledge on them might result in more efficient primality tests.

2. Properties of the Carmichael Numbers

Carmichael numbers are the composite numbers satisfying Fermat's little theorem. From the Chinese remainder theorem and the properties of primitive roots, we can obtain a characterization of these numbers.

Lemma 2. *(Chinese remainder theorem [4]) Let $n_1, n_2, ..., n_r$ be pairwise relatively prime positive integers. Then the system of congruences:*

$$\begin{aligned} x &\equiv a_1 \,(mod\, n_1) \\ x &\equiv a_2 \,(mod\, n_2) \\ &\vdots \\ x &\equiv a_r \,(mod\, n_r) \end{aligned} \tag{2}$$

has a unique solution modulo $N = n_1 n_2 ... n_r$.

Definition 3. *(Euler's totient function [5]) For a given integer n, $\phi(n)$ is the number of positive integers up to and relatively prime to n. For example, $1, 3, 7, 9$ are all the numbers up to and relatively prime to 10 and consequently, $\phi(10) = 4$.*

Definition 4. *(Primitive roots [6]) A number a is a primitive root modulo n if there exists an integer k for any value of b relatively prime to n that satisfies $a^k \equiv b \,(mod\, n)$. For example, with $3^1 = 3 \equiv 3 \,(mod\, 10)$, $3^2 = 9 \equiv 9 \,(mod\, 10)$, $3^3 = 27 \equiv 7 \,(mod\, 10)$ and $3^4 = 81 \equiv 1 \,(mod\, 10)$, we have that 3 is a primitive root modulo n, since $1, 3, 7, 9$ are all the relatively prime numbers up to 10.*

Lemma 5. [5],[6],[7] *If p is an odd prime and $k \geq 1$, there must exist a primitive root a modulo p^k. Furthermore, we have that*

$$(i)\, a^{\phi(p^k)} \equiv 1 \,(mod\, p^k) \quad (ii)\, \phi(p^k) = p^{k-1}(p-1)$$

From this we can derive the traditional characterization of Carmichael numbers known as Korselt's criterion:

Theorem 6. *(Korselt's criterion) A composite integer $m > 2$ is a Carmichael number if and only if m is square-free and for every prime p dividing m:*

$$p - 1 \,|\, m - 1 \tag{3}$$

Proof. This proof is inspired by the proof given in [8]. We start off by showing that a Carmichael number is odd. This can be proven by contradiction if we assume that $m = 2k$ is a Carmichael number, where k is an integer. The definition of Carmichael numbers then gives us the congruence $b^{2k-1} \equiv 1 \,(mod\, 2k)$. Setting $b = 2k-1$, which is coprime to $2k$, we obtain the congruence $(2k-1)^{2k-1} \equiv (-1)^{2k-1} \equiv -1 \equiv 1 \,(mod\, 2k)$. This congruence only holds for $k = 1$ and since $2k = 2$ isn't a composite number, we get a contradiction. Thus, all the prime factors of a Carmichael number are odd and Lemma 5 can be used.

We start off by writing an arbitrary Carmichael number of d prime factors as $m = \prod_{i=1}^{d} p_i^{k_i}$. Since the factors p_i are prime, we can conclude from Lemma 5 that there must exist a primitive root a_i modulo $p_i^{k_i}$ for each i. From the Chinese remainder theorem (Lemma 2) we know that there also must exist an a such that $a \equiv a_i \,(mod\, p_i^{k_i})$ for each i. a must not be divisible by any p_i for a_i to be primitive roots. Therefore, a is coprime to m and since m is a

Carmichael number, the congruence $a^{m-1} \equiv 1 \,(mod\, m)$ holds. Now, by reducing this congruence, we have that $a_i^{m-1} \equiv 1 \,(mod\, p_i^{k_i})$. Comparing this to Lemma 5 (i), we see that $\phi(p_i^{k_i})$ must divide $m-1$ and from Lemma 5 (ii) we get that $p_i^{k_i-1}(p_i-1) \mid m-1$. This only holds for $k_i = 1$, as p_i is a factor of m and therefore cannot divide $m-1$, proving that m is squarefree. Furthermore, this gives us that $p_i - 1 \mid m-1$ for each i. Therefore, if a number is a Carmichael number it is implied that it satisfies Korselt's criterion.

To obtain an equivalence, we examine a composite, squarefree number m satisfying Korselt's criterion. Now put an integer a prime to m. Then it follows that every prime factor p also is prime to a and Fermat's little theorem gives us $a^{p-1} \equiv 1 \,(mod\, p)$. For any integer s we have that $a^{s(p-1)} \equiv 1^s \,(mod\, p)$. Since $p-1 \mid m-1$, it follows that $a^{m-1} \equiv 1 \,(mod\, p)$. Furthermore, since this holds for all factors p of m, the congruence $a^{m-1} \equiv 1 \,(mod\, m)$ is obtained. □

This characterization was discovered by Alwin Korselt in 1899. As an example, we can check the numbers 147, 231 and 561. Factoring 147, we obtain the prime factors $\{3, 7, 7\}$ and the condition that the number must be squarefree allows us to determine that 147 is not a Carmichael number. For 231 we obtain the factors $\{3, 7, 11\}$ and can conclude that 231 is squarefree. For the prime factor 7 however, we have that $7-1 \nmid 231-1$ and consequently 231 does not satisfy Korselt's criterion. Finally, 561 is squarefree, as it has the factors $\{3, 11, 17\}$. Checking for all the factors, we get that $3-1 \mid 561-1$, $11-1 \mid 561-1$ and $17-1 \mid 561-1$ and 561 is thus a Carmichael number.

Unaware of the results of Korselt, Robert Carmichael went on to present the first of these numbers in 1910 [9], hence their name. In that paper, as well as a follow-up paper from 1912 [10], he showed some of the properties of these numbers:

Theorem 7. *Every Carmichael number m is odd, squarefree and comprised of three or more factors. If p and q are prime divisors of m, then*

$$(i)\ p-1 \mid m-1 \quad (ii)\ p-1 \mid \frac{m}{p}-1 \quad (iii)\ p \nmid q-1 \tag{4}$$

It also follows that any prime divisor $p < \sqrt{m}$. A proof will be given in Proposition 11.

Václav Šimerka already wrote about the first seven numbers in a paper from 1885 [12], but his work would go unnoticed. The first seven Carmichael numbers are:

$$561, 1105, 1729, 2465, 2821, 6601, 8911$$

In 1994 W.R. Alford et. al. proved that there exists infinitely many Carmichael numbers [13], based on the work Paul Erdős.

3. The Kellner-Sondow Characterization

Exploring Carmichael numbers in the context of p-adic theory, Kellner and Sondow [14] were able to give a new characterization for the Carmichael

numbers. They first defined $s_p(m)$ as the sum of the digits of m when written in base p. For example, $s_5(106) = 6$ can be obtained by writing $106 = (411)_5$ and then summing the digits $4 + 1 + 1 = 6$. Using this function, they were able to characterize the Carmichael numbers as follows:

Theorem 8. *An integer $m > 1$ is a Carmichael number if and only if m is squarefree and for every prime p dividing m:*

$$s_p(m) \geq p \quad and \quad s_p(m) \equiv 1 \ (mod\, p-1) \tag{5}$$

This characterization was obtained without assuming compositeness and using the properties of the s_p function. Kellner and Sondow defined the set S as the squarefree integers m greater than 1, for which every prime divisor p satisfies $s_p(m) \geq p$. They then managed to prove that the Carmichael numbers were a subset of S.

Similarly to the example for Korselt's criterion (Theorem 6), we examine the numbers $231 = 3 \cdot 7 \cdot 11$ and $561 = 3 \cdot 11 \cdot 17$ again. Since $231 = (450)_7$, we get that $s_7(231) = 9$. As $9 \not\equiv 1 \ (mod\, 7-1)$ we can confirm that 231 is not a Carmichael number. For 561 however, we can obtain the digit sums $s_3(561) = 7$, $s_{11}(561) = 11$ and $s_{17}(561) = 17$. These all satisfy the congruence in Theorem 8 and 561 is once again confirmed to be a Carmichael number.

To effectively make use of the s_p function we first describe an arbitrary integer N as $N = a_k b^k + a_{k-1} b^{k-1} + ... + a_1 b + a_0$, where b is the desired base. The digits for $(N)_b$ are then the coefficients $\{a_k, a_{k-1}, ..., a_1, a_0\}$ and $s_p = \sum_{i=0}^{k} a_i$. For a factor p of a Carmichael number m, we note that m can be written as $m = a_k p^k + a_{k-1} p^{k-1} + ... + a_1 p$, i.e. $a_0 = 0$, as p must divide m. Since m is squarefree, we get that $a_1 > 0$ and setting $m_p = m/p$ we get that $s_p(m) = s_p(m_p)$.

We can now examine the similarity between the new characterization and Korselt's criterion. Using the description above, we can write that $m - s_p(m) = \sum_{i=1}^{k} a_i p^i - \sum_{i=1}^{k} a_i$. Combining the sums, we get that $m - s_p(m) = \sum_{i=1}^{k} a_i (p^i - 1)$. For any positive integer x, $p^x - 1 = (p-1) \sum_{i=0}^{x-1} p^i$, i.e. a multiple of $p-1$. Thus, $p - 1 \mid m - s_p(m) = (m-1) - (s_p(m) - 1)$. Korselt's criterion states that $p - 1 \mid m - 1$ and, consequently, $p - 1 \mid s_p(m) - 1$, which matches Theorem 8. For a more detailed analysis, see Section 4 of [14, p. 8].

Another consequence of this description can be seen, when studying the largest factor r of a Carmichael number. If m_r is the product of the remaining factors, i.e. $m_r = m/r$, and d is the total number of factors, then $r^{d-1} > m_r > r$. The upper limit comes naturally from r being the largest factor. For the lower limit, assume that $m_r \leq r$. $(m_r)_r$ is then represented by a single value and as we have for any digit a_k that $a_k < r$, $s_p(m_r)$ becomes less than r and is thus unable to satisfy Theorem 8. From this we can write m_r as $m_r = a_{d-1} r^{d-2} + a_{d-2} r^{d-3} + ... + a_1$. $(m_r)_r$ and, consequently, $(m)_r$ have a number of digits less than or equal to $d-1$. Setting $d = 2$ would yield one digit and thus not satisfy Theorem 8, proving that $d \geq 3$.

Other properties of a Carmichael number m, such as $2 \nmid m$ and $p < \sqrt{m}$ for any factor p of m, can also be derived using the new characterization. More-

over, the property given in Theorem 7 (ii) can also be proved using this new characterization:

Corollary 9. *If p is a factor of a Carmichael number m, with m_p being the product of the remaining factors, then*

$$p-1 \mid m_p - 1 \tag{6}$$

Proof. To prove Equation 6, we start off with the simplest example before generalizing for any p and m_p. If the number of factors $d = 3$ and r is the largest factor, then $(m_r)_r$ has two digits a_2, a_1 and m_r can be written as $m_r = a_2 r + a_1$. Describing $s_r(m_r)$ in terms of a_1, we get

$$s_r(m_r) = \frac{m_r - a_1}{r} + a_1 \tag{7}$$

From Theorem 8 we have that $r-1 \mid \frac{m_r - a_1}{r} + a_1 - 1$. As multiplication with an integer doesn't destroy divisibility, we multiply with r to get that $r-1 \mid m_r - r + (r-1)a_1$. Adding and subtracting by 1, we get $r-1 \mid m_r - 1 - (r-1) + (r-1)a_1$, with the last two terms being obvious multiples of $r-1$. This means that $m_r - 1$ also has to be a multiple and thus $r-1 \mid m_r - 1$.

Generalizing this for any m_p yielding n digits when written in base p, we can write $s_p(m_p)$ in a similar fashion:

$$s_p(m_p) = \frac{m_p - \sum_{i=1}^{n-1} a_i p^{i-1}}{p^{n-1}} + \sum_{i=1}^{n-1} a_i \tag{8}$$

where $a_1, a_2, ...$ are the digits of m_p in the base of p. Following the same procedure, we end up with $p-1 \mid m_p - 1 - (p^{n-1} - 1) + \sum_{i=1}^{n-1} (p^{n-i} - 1) p^{i-1} a_i$. Since for any positive integer x, we have that $p^x - 1 = (p-1) \sum_{i=0}^{x-1} p^i$, we can disregard all the terms with coefficients of this shape. This leaves us with $p-1 \mid m_p - 1$ once again. □

This property can be very useful in the computation of Carmichael numbers and is used in both the algorithms appearing in the following section. Kellner and Sondow also gave another relationship with clear applications for the computation, namely a sharp upper limit for the prime factors:

Theorem 10. *For any prime p dividing a Carmichael number m:*

$$p \leq \sqrt{\frac{17m}{33}} \tag{9}$$

4. Making an algorithm

As Carmichael numbers are characterized in terms of their prime factors, the challenge of creating an algorithm that generates these numbers seems to be how to obtain the factors and check them in as few steps as possible. To

avoid time consuming primality tests, a precomputed list of primes can be useful when checking these factors. Obtaining the factors first and then constructing the Carmichael number has the advantage that a factorization process can be avoided. No efficient integer factorization algorithm is known, although it hasn't been proved that no such algorithm exists [15].

My work was based on the algorithm used by Pinch to generate all the Carmichael numbers up to 10^{15}. Pinch constructed the algorithm using the following three propositions [16]:

Proposition 11. *Let m be a Carmichael number less than X, with d factors arranged in increasing order $p_1 < p_2 < ... < p_d$.*
(i) Let $n < d$ and put $P = \prod_{i=1}^{n} p_i$. Then $p_{n+1} < (X/P)^{1/(d-n)}$ and p_{n+1} is prime to $p_i - 1$ for all $i \leq n$.
(ii) Put $m_r = \prod_{i=1}^{d-1} p_i$ and $L = lcm\{p_1 - 1, ..., p_{d-1} - 1\}$. Then, for the largest factor r, $m_r r \equiv 1 \,(mod\, L)$ and $r - 1$ divides $m_r - 1$.
(iii) Each prime factor p satisfies $p < \sqrt{m} < \sqrt{X}$.

Proof. To prove part (i), we first note that for any integer n, m has $d-n$ factors larger than or equal to p_{n+1}. This gives us the inequality $P \cdot p_{n+1}^{d-n} < m$ and, consequently, $p_{n+1} < (X/P)^{1/(d-n)}$. Since p_{n+1} is a prime number larger than $p_i - 1$ for all $i \leq n$, it follows that p_{n+1} is prime to $p_i - 1$. Part (ii) follows directly from Korselt's criterion and $r - 1 \mid m_r - 1$ is proven in Corollary 9. For part (iii) we also use the results of Corollary 9. $p - 1 \mid m_p - 1$ implies that $p \leq m/p$ and the condition that m is squarefree gives a strict inequality, as $p = m/p \Leftrightarrow m = p^2$. Therefore, it follows that for any factor p of a Carmichael number m, $p < \sqrt{m} < \sqrt{X}$ [11]. □

Proposition 12. *Let $P = \prod_{i=1}^{d-2} p_i$. There are integers $2 \leq D < P < C$ such that, putting $\Delta = CD - P^2$, we have*

$$q = \frac{(P-1)(P+D)}{\Delta} + 1,$$
$$r = \frac{(P-1)(P+C)}{\Delta} + 1,$$
$$P^2 < CD < P^2(\frac{p_{d-2}+3}{p_{d-2}+1})$$

Where q and r are the second largest and the largest prime factor, respectively.

Proposition 13. *Let $P = \prod_{i=1}^{d-2} p_i$. Then*

$$q < 2P^2$$
$$r < P^3$$

Proof. To prove the inequalities we use Proposition 12. Putting $\Delta = 1$, $d < P$ and substituting into the expression for q we get that $q < (P-1)(2P)+1 < 2P^2$.

For the second inequality we use the inequality of Proposition 13. Putting $D \geq 2$ and $p_{d-2} \geq 3$, have that $C \leq 3P^2/4$. Using this in the expression for r, we obtain $r < P^3$. □

At the first stage of Pinch's algorithm, successive lists of primes $p_1, ..., p_{d-2}$ are generated, following the inequality of Proposition 11 (i). From these primes $P = \prod_{i=1}^{d-2} p_i$ is then calculated and depending on the size of P, two different algorithms are used to obtain q and r. Pseudocode representation:

Algorithm 1 Pinch's algorithm

Use Proposition 11 (i) to generate lists of primes

for all lists of primes **do**

  $P \leftarrow$ the product of the primes

  if P is small **then**

    SMALLP(P)

  end if

  if P is large **then**

    LARGEP(P)

  end if

end for

For smaller values of P, Proposition 12 is used: first by looping over all D within $2 \leq D < P$, then all C permitted by the inequality for Proposition 12 (iii). If the resulting q and r are prime numbers satisfying Korselt's criterion, then the product Pqr is a Carmichael number. Pseudocode representation:

Algorithm 2 For Smaller P

function SMALLP(P)

  for $D \leftarrow 2$ to P **do**

    for $C \leftarrow P^2/D$ to $P^2(\frac{p_{d-2}+3}{p_{d-2}+1})/D$ **do**

      Calculate q and r according to Proposition 12

      if prime and satisfying Korselt's criterion **then**

        add Pqr to list of Carmichael numbers

      end if

    end for

  end for

end function

For larger values of P, q permitted by Proposition 11 (i) and 13 (i) are instead

looped over. For such q, $m_r = Pq$ and $L = lcm\{p_1 - 1, ..., p_{d-2} - 1, q - 1\}$ are calculated and the possible r are computed in two distinct loops.

For smaller r, a value P' that satisfies $m_r P' \equiv 1 \,(mod\, L)$ is computed. For numbers congruent to $P' \,(mod\, L)$ and larger than q, a primality check and Korselt's criterion are used to determine if r is a factor of a Carmichael number.

Larger r are obtained by running over the small factors f of $m_r - 1$ and computing $(m_r - 1)/f + 1$. If the resulting number is a prime r satisfying $m_r r \equiv 1 \,(mod\, L)$, then it is a factor of a Carmichael number. Pseudocode representation:

Algorithm 3 For Larger P

```
function LARGEP(P)
    Obtain q permitted by Propositions 11 (i) and 13 (i)
    for all permitted q do
        m_r ← Pq
        for small r larger than q do
            Find P' were m_r P' ≡ 1 (mod L)
            if r is prime, r ≡ P' (mod L) and r satisfies Korselt's criterion then
                add Pqr to list of Carmichael numbers
            end if
        end for

        for small f do
            r ← (m_r − 1)/f + 1
            if integer, prime and m_r r ≡ 1 (mod L) then
                add Pqr to list of Carmichael numbers
            end if
        end for
    end for
end function
```

Pinch's algorithm effectively limits the numbers checked for the last two prime factors, by expressing them in terms of P. Moreover, the algorithm is split into smaller algorithms at two points depending on the size of P and r, respectively. This allows the values of q and r to be computed appropriately for the given size, however finding a suitable range where to consider a value large or small requires some testing. The process where successive lists of primes are created can also be time consuming for certain X and for $X = 10^6$ it took up roughly 41% of the total computing time. In the creation of these lists, it is also necessary to know the maximal number of factors for a given X to avoid

unnecessarily large lists.

With this in mind, I created my own algorithm. The idea was to start from the largest factor and then generate the lower factors to obtain the Carmichael numbers. This required a factorization process, which was mentioned before to be a slow process for large arbitrary integers. The speed of the algorithm was therefore dependent on how effectively the conditions given from the properties of Carmichael numbers could limit this process. The new findings of Kellner and Sondow were also taken into consideration and the sharp estimate from Theorem 10 allowed for a shorter list of primes to be examined. Other helpful conditions were unfortunately not found and the characterization itself was shown to be less efficient than Korselt's criterion for testing.

In a run time comparison, I let two functions examine lists of a random number of randomized prime factors. The first function used Korselt's criterion to determine if the factors constituted a Carmichael number and the second one used the Kellner-Sondow characterization. There was a total of 10^5 lists and both functions examined the same lists. The first function completed the task with an average of $2.0075s$ and the second in $2.7947s$, meaning that the run time for checking with the Kellner-Sondow characterization was on average roughly 39% slower. That being said, the choice of testing method had a minimal effect on the speed of the algorithm, with the testing taking up a maximum of roughly 5% or 7% of the total run time of my algorithm for X up to 10^{10}, depending on the method.

Using Theorem 10 we start by looping over every prime r up to $\sqrt{\frac{17X}{33}}$, where X is the desired upper limit for the Carmichael numbers. Now we can loop for integers f larger than 1 and use Theorem 7 (ii) to obtain m_r as $m_r = f(r-1)+1$. As for the upper limit of m_r, the smallest value of X/r and r^{D-1} is chosen, where D is the maximum number of factors for a Carmichael number within X. These m_r are then factorized. These factors must be prime numbers satisfying Korselt's criterion and the number of factors must be less than or equal to $D-1$. If a square or a factor not satisfying this is found, we immediately move on to the following m_r, to shorten the factorization process. The factorization process grows to be the most lengthy process for higher m_p and shortening this process has great significance for the efficiency of the algorithm. If all factors satisfy the criterion, however, these are multiplied with r to obtain a Carmichael number.

Algorithm 4 is a pseudocode presentation of the procedure:

Algorithm 4 My algorithm

for every permitted prime r **do**
  $M \leftarrow$ the lower value of r^{D-1} and X/r
  for f where $r < m_r < M$ **do**
    $m_r \leftarrow f(r-1)+1$
    for every possible factor of m_r **do**
      if squares, factors not satisfying Korselt's criterion or more than $D-1$ factors are found **then**
        Break loop
      end if
    end for
    if all factors satisfy Korselt's criterion **then**
      $m_r r$ is a Carmichael number
    end if
  end for
end for

Both of the algorithms, written in Python code, can be found here.

5. Results

The new algorithm had less of a dependency on the number of factors, as the factorization process didn't assume a certain number of factors. However, since the upper limit X/r is large for $r << X$, the secondary upper limit p^{D-1} was introduced, with D being the maximum number of factors, and the factorization process was stopped after $D-1$ factors to avoid superfluous calculations. From the results in Pinch's article, we can see how many prime factors are needed, for a given upper bound X.

Table 1. Number of Carmichael numbers with d prime factors up to 10^{15} [16].

	d							
$log_{10}X$	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**Total**
3	1	0	0	0	0	0	0	1
4	7	0	0	0	0	0	0	7
5	12	4	0	0	0	0	0	16
6	23	19	1	0	0	0	0	43
7	47	55	3	0	0	0	0	105
8	84	144	27	0	0	0	0	255
9	172	314	146	14	0	0	0	646
10	335	619	492	99	2	0	0	1547
11	590	1179	1336	459	41	0	0	3605
12	1000	2102	3156	1714	262	7	0	8241
13	1858	3639	7082	5270	1340	89	1	19279
14	3284	6042	14938	14401	5359	655	27	44706
15	6083	9938	29282	36907	19210	3622	170	10212

The comparison was performed by testing the speed of the algorithms for different values of X. To more accurately compare the time of the computation Pinch's algorithm was modified to have the sharp upper limit $\sqrt{\frac{17X}{33}}$, given by Theorem 10.

TABLE 2. Run time comparison in seconds between Pinch's algorithm (**Pinch**) and the new one (**New**) for Carmichael numbers up to X.

$log_{10}X$	**5**	**6**	**7**	**8**
Pinch	0.140625	0.484375	3.1875	29.078125
New	0.046875	0.59375	3.15625	21.609375

In Table 2, the new algorithm seems to be slightly faster. Although the algorithms are of comparable speed, the new algorithm has the advantage that there is no split in computation methods and therefore there is no need to find were to put this split. However due to time constraints, larger values of X weren't checked and it is possible that Pinch's algorithm is more efficient for larger Carmichael numbers.

While the new algorithm may allow for quick calculations of Carmichael number, the methods used also come with limitations. The Carmichael numbers for a certain number of factors cannot be computed, as the number of factors of m are unknown before the factorization. Furthermore, considering Table 3, we can see that for larger m the factorization process grows quickly. The algorithm used a rudimentary factorization method and more efficient factorization methods may be used, but factorizing an integer is a problem generally considered to be a slow process [15]. This means that for larger values of m, the efficiency of the algorithm might drop off and in that case, a supplementary method would be necessary.

TABLE 3. Comparison of the run times for the factorization (Factor) and the total process (Total), as well as their ratio for a given upper limit X.

$log_{10}X$	**Factor**	**Total**	**Factor/Total**
5	0.015625	0.046875	0.33
6	0.15625	0.59375	0.26
7	1.40625	3.15625	0.39
8	13.171875	21.609375	0.52
9	148.09375	294.890625	0.50
10	1917.296875	3872.5625	0.50

Examining Table 3, we can see that the relative factorization time shows a tendency to increase for larger numbers as expected. Curiously, the relative time dips for certain X and it seems to be specifically those X that entails an increase in D.

Finally, it is worth noting that the programs were written in Python, which is an interpreted language. Interpreted languages are famously slower [17], but with the use of the NumPy [18] module which is written in C, computation can be significantly sped up. This module was purposefully avoided, however, for the sake of comparison and with the use of NumPy, the run time could have most likely been shortened for both the algorithms.

6. Acknowledgements

I would like to express my deepest appreciation to Anna Torstensson for helpful discussions, feedback and suggestions.

7. References

[1] Agrawal, M., Kayal, N. and Saxena, N. (2004) *Primes is in P*. Annals of Mathematics. 160 (2): 781-793. Available at: primality_v6.pdf (Accessed 2 June 2022).

[2] Conway, J. H. and Guy, R. (1996) *The Book of Numbers*. New York: Springer-Verlag, p. 141-143.

[3] Korselt, A. (1899) *Problème Chinois*. L'intermédiaire des mathématiciens. 6: p. 142-144. Available at: Problème_chinois.pdf (Accessed 2 June 2022).

[4] Lac, J. H. (2008) *Chinese remainder theorem and its applications*. Theses Digitization Project. 3373. Available at: Chinese remainder theorem and its applications (Accessed 12 June 2022).

[5] Wolfram Mathworld (2014) Available at: mathworld.wolfram.com (Accessed 12 June 2022).

[6] Encyclopedia of Mathematics (2014) Available at: encyclopediaofmath.org (Accessed 12 June 2022).

[7] Gamboa, R., Gamboa, W. (2022) *Prime Numbers Have Primitive Roots*. arXiv:2205.11694v1 [cs.LO]. EPTCS 359, 9-18. Available at: 2205.11694.pdf (Accessed 12 June 2022).

[8] Ma, D. (2013) Available at: exploringnumbertheory (Accessed 12 June 2022).

[9] Carmichael, R. D. (1910) *Note on a New Number Theory Function*. Bulletin of the American Mathematical Society. 16 (5): 232–238. Available at: S0002-9904-1910-01892-9.pdf (Accessed 2 June 2022).

[10] Carmichael, R. D. (1912) *On Composite Numbers P Which Satisfy the Fermat Congruence*. The American Mathematical Monthly. 19 (2): 22-27. Available at: jstor.org (Accessed 2 June 2022).

[11] Conrad, K. (2016) *Carmichael number and Korselt's criterion*. Available at: carmichaelkorselt.pdf (Accessed 2 June 2022).

[12] Šimerka, V. (1885) *Zbytky z arithinetické posloupnosti (On the remainders of an arithmetic progression)*. Časopis Pro Pěstování Matematiky a Fysiky. 14 (5): 221–225. Available at: CasPestMatFys_014-1885-5_3.pdf (Accessed 2 June 2022).

[13] Alford, W. R., Granville, A. and Pomerance, C. (1994) *There are Infinitely Many Carmichael Numbers*. Annals of Mathematics. 140 (3): 703–722. Available at: paper95.pdf (Accessed 2 June 2022).

[14] Kellner, B. C. and Sondow, J. (2021) {On Carmichael and Polygonal Numbers, Bernoulli Polynomials, and Sums of Base-p Digits. arXiv:1902.10672v2 [math.NT] Integers 21 (2021), Article A52: 1-21. Available at: 1902.10672.pdf (Accessed 2 June 2022).

[15] Krantz, S. G., (2011) *The Proof is in the Pudding: The Changing Nature of Mathematical Proof*. New York: Springer, p. 203.

[16] Pinch, R. G. E. (1993) *The Carmichael Numbers up to* 10^{15}. Mathematics of Computation. 61 (203): 381-391. Available at: S0025-5718-1993-1202611-7.pdf (Accessed 2 June 2022).

[17] Programiz (2016) Available at: https://www.programiz.com/article/difference-compiler-interpreter (Accessed 2 June 2022).

[18] NumPy (2020) Available at: numpy.org (Accessed 2 June 2022).