

## TWO ALGORITHMS TO FIND PRIMES IN PATTERNS

JONATHAN P. SORENSON AND JONATHAN WEBSTER

ABSTRACT. Let  $k \geq 1$  be an integer, and let  $(f_1(x), \dots, f_k(x))$  be  $k$  admissible linear polynomials over the integers. We present two algorithms that find all integers  $x$  where  $\max\{f_i(x)\} \leq n$  and all the  $f_i(x)$  are prime.

- Our first algorithm takes at most  $O(nk/(\log \log n)^k)$  arithmetic operations using  $O(k\sqrt{n})$  space.
- Our second algorithm takes slightly more time,  $O(nk/(\log \log n)^{k-1})$  arithmetic operations, but uses only  $\exp O(\log n / \log \log n)$  space. This result is unconditional for  $k > 6$ ; for  $2 < k \leq 6$ , the proof of its running time, but not the correctness of its output, relies on an unproven but reasonable conjecture due to Bach and Huelsbergen.

We are unaware of any previous complexity results for this problem beyond the use of a prime sieve.

We also implemented several parallel versions of our second algorithm to show it is viable in practice. In particular, we found some new Cunningham chains of length 15, and we found all quadruplet primes up to  $10^{17}$ .

## 1. INTRODUCTION

Mathematicians have long been interested in prime numbers and how they appear in patterns. (See, for example, [12, ch. A].) In this paper, we are interested in the complexity of the following algorithmic problem:

Given a pattern and a bound  $n$ , find all primes  $\leq n$  that fit the pattern.

To address this, first we will discuss and define a pattern of primes, then we will look at what is known about the distribution of primes in patterns to see what we can reasonably expect for the complexity of this problem, and finally we will discuss previous work and state our new results.

**1.1. Prime Patterns.** Perhaps the simplest of patterns of primes are the twin primes, which satisfy the pattern  $(x, x+2)$  where both  $x$  and  $x+2$  are prime. Examples include 59,61 and 101,103.

We can, of course, generalize this to larger patterns. For example, prime quadruplets have the form  $(x, x+2, x+6, x+8)$ , and examples include 11,13,17,19 and 1481,1483,1487,1489.

Larger patterns of primes are called *prime  $k$ -tuples*. If the  $k$ -tuple has the smallest possible difference between its first and last primes (its *diameter*),

---

1991 *Mathematics Subject Classification.* 11A41,11Y11,11Y16,68Q25.

it is also called a *prime constellation*. See, for example, [7, §1.2.2] or [25, ch. 3].

Sophie Germain studied the pattern  $(x, 2x + 1)$ , which was later generalized to *Cunningham chains* of two kinds. Chains of the *first kind* have the pattern  $(x, 2x + 1, 4x + 3, 8x + 7, \dots)$ , and chains of the *second kind* have the pattern  $(x, 2x - 1, 4x - 3, 8x - 7, \dots)$ .

Chernick [6] showed that any prime pattern of the form  $(6x + 1, 12x + 1, 18x + 1)$ , which is admissible, gives a Carmichael number composed of the product of these three primes.

Let  $k > 0$  be an integer. A *prime pattern* of size  $k$  is a list of  $k$  linear polynomials over the integers with positive leading coefficients,  $(f_1(x), \dots, f_k(x))$ . A pattern of size  $k$  is *admissible* if for every prime  $p \leq k$ , there is an integer  $x$  such that  $p$  does not divide any of the  $f_i(x)$ .

We restrict our notion of pattern to linear polynomials in this paper.

**1.2. The Distribution of Primes in Patterns.** The unproven twin prime conjecture states there are infinitely many twin primes. Yitang Zhang [31] recently showed that there is a positive integer  $h < \infty$  such that the pattern  $(x, x + h)$  is satisfied by infinitely many primes.

The Hardy-Littlewood  $k$ -tuple conjecture [14] implies that each pattern, with leading coefficients of 1, that is *admissible*, will be satisfied by primes infinitely often. Further, the conjecture implies that the number of primes  $\leq n$  in such a pattern of length  $k$  is roughly proportional to  $n/(\log n)^k$ .

For twin primes, then, the Hardy-Littlewood conjecture gives an estimate of

$$2C_2 \frac{n}{(\log n)^2}$$

for the number of twin primes  $\leq n$ , where  $C_2 \approx 0.6601\dots$  is the twin primes constant. Brun's theorem gives an  $O(n/(\log n)^2)$  upper bound for the number of twin primes  $\leq n$ .

Dickson's conjecture [8] states that there are infinitely many primes satisfying any fixed admissible pattern, even with leading coefficients  $\geq 1$ . Thus, it applies to Cunningham chains as well.

We have the following upper bound, due to Halberstam and Richert [13, Theorem 2.4].

**Lemma 1.** *Given an admissible pattern  $(f_1(x), \dots, f_k(x))$  of length  $k$ , the number of integers  $x$  such that the  $f_i(x)$  are all simultaneously prime and  $\max\{f_i(x)\} \leq n$  is at most  $O(n/(\log n)^k)$ .*

Here the implied constant of the big- $O$  can depend on  $k$ , but does not depend on the coefficients of the  $f_i$ .

**1.3. Previous Work.** For previous computational work on finding primes in patterns, see the work of Günter Löh [19] and Tony Forbes [9]. Both

authors described their algorithms, but gave no runtime analysis. In particular, Forbes outlined an odometer-like data structure that mimicks the wheel data structure we employ.

But to get a complexity measure, the best previous work is to find *all* primes  $\leq n$ , and then scan the resulting list of primes to see how many match the pattern. In other words, this is basically a prime number sieve.

The Atkin-Bernstein sieve [2] does this using at most  $O(n/\log \log n)$  arithmetic operations and  $\sqrt{n}$  space. Galway [10] showed how to reduce space further to roughly  $n^{1/3}$ , but this version could not use the wheel data structure and requires  $O(n)$  arithmetic operations.

We follow the convention of not charging for space used by the output of the algorithm.

Of course, by the prime number theorem, there are only about  $n/\log n$  primes  $\leq n$ , so the current best prime sieves use  $\log n/\log \log n$  arithmetic operations per prime on average. We do not know if anything smaller is possible. Applying this average cost to the results of Lemma 1, we can hope for an algorithm that takes  $O(n/(\log \log n)^k)$  arithmetic operations to find all primes in a fixed pattern of length  $k$ .

In essence, this is what we prove as our main result.

**1.4. New Results.** Our contribution is the following.

**Theorem 1.** *Given a list of  $k$  distinct linear polynomials over the integers, with positive leading coefficients,  $(f_1(x), \dots, f_k(x))$  (the pattern), and a search bound  $n$ , there is an algorithm to find all integers  $x$  such that  $\max\{f_i(x)\} \leq n$  and all the  $f_i(x)$  are prime. This algorithm uses at most  $O(nk/(\log \log n)^k)$  arithmetic operations and  $O(k\sqrt{n})$  bits of space.*

This algorithm extends the Atkin-Bernstein prime sieve with our space-saving wheel sieve. See [27, 28, 29].

The  $\sqrt{n}$  space needed by this algorithm limits its practicality. By replacing the use of the Atkin-Bernstein sieve with the sieve of Eratosthenes combined with prime tests, we can greatly reduce the need for space.

**Theorem 2.** *Given a list of  $k > 6$  distinct linear polynomials over the integers, with positive leading coefficients,  $(f_1(x), \dots, f_k(x))$  (the pattern), and a search bound  $n$ , there is an algorithm to find all integers  $x$  such that  $\max\{f_i(x)\} \leq n$  and all the  $f_i(x)$  are prime. This algorithm uses at most  $O(nk/(\log \log n)^{k-1})$  arithmetic operations and  $\exp O(\log n/\log \log n)$  bits of space.*

This second version requires  $k > 6$  to be unconditional. We use the sieve of Eratosthenes with primes up to a bound  $B = \exp O(\log n/\log \log n)$ , after which we apply a base-2 pseudoprime test and then a version of the AKS prime test [1] that takes  $(\log n)^{6+o(1)}$  time. The algorithm works for  $2 < k \leq 6$ , but to keep the cost of prime testing low enough, we replace the AKS prime test with the pseudosquares prime test of Lukes, Patterson,

and Williams [20]. This prime test takes only  $O((\log n)^2)$  time under the condition of an unproven but reasonable conjecture on the distribution of pseudosquares due to Bach and Huelsbergen [3]. Note that correctness does not rely on any unproven conjectures.

The *pseudosquare*  $L_p$  is the smallest positive integer that is not a square,  $L_p \equiv 1 \pmod{8}$ , and for every odd prime  $q \leq p$ , we have  $(L_p/q) = 1$ , where  $(x/y)$  is the Legendre symbol.

**Conjecture 1** (Bach and Huelsbergen [3]). *Let  $L_p$  be the largest pseudosquare  $\leq n$ . Then  $p = O(\log n \log \log n)$ .*

**Theorem 3.** *Given a list of  $k > 2$  distinct linear polynomials over the integers, with positive leading coefficients,  $(f_1(x), \dots, f_k(x))$  (the pattern), and a search bound  $n$ , there is an algorithm to find all integers  $x$  such that  $\max\{f_i(x)\} \leq n$  and all the  $f_i(x)$  are prime. If Conjecture 1 is true, this algorithm uses at most  $O(nk/(\log \log n)^{k-1})$  arithmetic operations and  $\exp O(\log n/\log \log n)$  bits of space.*

We performed a few computations with this last version of our algorithm to show its practicality. A couple of these computations are new.

The rest of this paper is organized as follows. In §2 we present our proof of Theorem 1, including our model of computation in §2.1, a description of our first algorithm in §2.2, and its running time analysis in §2.3. In §3 we discuss our second algorithm in §3.1, and its analysis in §3.2, thereby proving Theorems 2 and 3. We present our computational results in §4, including our work on twin primes in §4.1, our work on prime quadruplets in §4.2, and our results on Cunningham chains in §4.3. We wrap up with a discussion of possible future work in §5.

## 2. THEORY

**2.1. Model of Computation.** Our model of computation is a standard random access machine with infinite, direct-access memory. Memory can be addressed at the bit level or at the word level, and the word size is  $\Theta(\log n)$  bits, if  $n$  is the input. Arithmetic operations on integers of  $O(\log n)$  bits take constant time, as do memory/array accesses, comparisons, and other basic operations.

We count space used in bits, and we do not include the size of the output.

**2.2. The Algorithm.** In this section we present the version of our algorithm with the smallest running time; we perform the analysis in the next section.

The input to the algorithm is the search bound  $n$  and the pattern, which consists of the value of  $k$  and the list of linear polynomials  $(f_1(x), \dots, f_k(x))$ . We write  $a_i$  for the multiplier and  $b_i$  for the offset for each form  $f_i$ . For simplicity, we often assume that  $a_1 = 1$  and  $b_1 = 0$ , but this convenience is not required to obtain our complexity bound. So for example, for Cunningham

chains of the first kind we would have  $a_1 = 1, a_2 = 2, a_3 = 4, \dots, a_k = 2^{k-1}$  and  $b_1 = 0, b_2 = 1, b_3 = 3, \dots, b_k = a_k - 1$ .

- (1) We begin by finding the list of primes up to  $\lfloor \sqrt{n} \rfloor$  and choosing a subset to be the *wheel primes*  $\mathcal{W}$ . Define  $W := \prod_{p \in \mathcal{W}} p$ . We put as many small primes into  $\mathcal{W}$  as possible, with the constraint that  $W \leq \sqrt{n}$ . If  $y$  is the largest prime in  $\mathcal{W}$ , we have

$$W = \prod_{p \in \mathcal{W}} p = \prod_{p \leq y} p = \exp \vartheta(y),$$

where  $\vartheta(x) = \sum_{p \leq x} \log p$  (see [15]), and so  $y \sim \vartheta(y) = \log W \sim (1/2) \log n$ . This implies  $\sqrt{n}/\log n \ll W \leq \sqrt{n}$ .

- (2) Next, we construct the wheel data structure so that it will generate all suitable residues modulo  $W$ .

The data structure is initialized with a list of pairwise coprime moduli, and for each such modulus  $m$ , a list of acceptable residues mod  $m$ , encoded as a bit vector of length  $m$ . The wheel modulus  $W$  is then the product of these moduli. Once initialized, the wheel has the ability to enumerate suitable residues modulo  $W$  in amortized constant time per residue. The residues do not appear in any particular order.

Therefore, for each prime  $p \in \mathcal{W}$  we compute a bit vector (`ones []`) that encodes the list of acceptable residues. For any integral  $x$ , we want  $f_i(x) = a_i x + b_i$  to not be divisible by  $p$ . So if  $p$  divides  $a_i x + b_i$ , or equivalently if  $x \equiv -b_i \cdot a_i^{-1} \pmod{p}$ , then  $x$ 's bit position must be a zero.

```
vector<bool> ones(p,1); // length p, all 1s to start
for(i=0; i<k; i++)
    ones[ (-b_i * a_i^{-1} mod p) ]=0;
```

Continuing the example above, for Cunningham chains of the first kind,  $p = 3$  gives the vector 001, and  $p = 7$  gives the vector 0010111.

We then construct the wheel data structure as described in [28, §4].

- (3) For each residue  $r \pmod{W}$  generated by the wheel, we sieve  $k$  arithmetic progressions for primes up to  $n$ ,  $f_i(r) = a_i r + b_i \pmod{W}$ , or  $(a_i r + b_i) + j \cdot a_i W$  for  $j = 0, \dots, \lfloor n/(a_i W) \rfloor$ .

We do this using the Atkin-Bernstein sieve. (See [2, §5] for how to use the sieve to find primes in an arithmetic progression.) This yields  $k$  bit vectors of length  $\leq n/W$  which are combined using a bitwise AND operation to obtain the bit positions for where the pattern is satisfied by primes.

**2.3. Complexity Analysis.** We'll look at the cost for each step of the algorithm.

- (1) We can use the Atkin-Bernstein sieve to find the primes up to  $\sqrt{n}$  in  $O(\sqrt{n}/\log \log n)$  arithmetic operations using  $n^{1/4}$  space.

- (2) Recall that the largest wheel prime is roughly  $(1/2)\log n$ . Constructing the bit vector `ones []` for one prime takes  $O(\log n)$  time to initially write all ones, and then  $O(k)$  time to mark the zeros. Summing over all wheel primes gives  $O((\log n)^2/\log \log n)$  operations.

From [28, Theorem 4.1] the total cost to build the wheel is  $O((\log n)^3)$  operations and it occupies  $O((\log n)^3/\log \log n)$  space.

- (3) The Atkin-Bernstein sieve finds all primes in an arithmetic progression in an interval of size  $\sqrt{n}$  or larger in time linear in the length of the interval using space proportional to  $\sqrt{n}$  [2, §5]. Therefore, sieving for primes takes  $O(n/W)$  operations for each of the  $k$  residue classes  $f_i(r) \bmod W$ , for a total of  $O(kn/W)$ . The cost to generate each value of  $r$  using the wheel is negligible in comparison. The space used is  $O(k\sqrt{n})$  bits.

Next we show that the total number of residues is asymptotic to  $W/(\log \log n)^k$ . For a pattern of size  $k$ , all but finitely many primes  $p$  will have  $p - k$  possible residues. So the total number of residues  $r \bmod W$  will be asymptotic to

$$\begin{aligned} \prod_{p \in \mathcal{W}} (p - k) &= W \prod_{p \in \mathcal{W}} \frac{p - k}{p} \\ &= W \prod_{p \in \mathcal{W}} \left(1 - \frac{k}{p}\right). \end{aligned}$$

By Bernoulli's inequality we have  $1 - k/p \leq (1 - 1/p)^k$ , so that

$$W \prod_{p \in \mathcal{W}} \left(1 - \frac{k}{p}\right) \leq W \prod_{p \in \mathcal{W}} \left(1 - \frac{1}{p}\right)^k.$$

As shown above, if  $y$  is the largest prime in  $\mathcal{W}$ , we have  $y \sim (1/2)\log n$ . Asymptotically,  $y \geq (1/10)\log n$  is a very safe underestimate. Applying Mertens's theorem we obtain

$$\begin{aligned} W \prod_{p \in \mathcal{W}} \left(1 - \frac{1}{p}\right)^k &\ll W \prod_{p \leq (1/10)\log n} \left(1 - \frac{1}{p}\right)^k \\ &\sim W \left(\frac{e^{-\gamma}}{\log \log n}\right)^k. \end{aligned}$$

Multiplying the sieving cost by the number of residues gives  $O(nk/(\log \log n)^k)$  operations.

We have proved Theorem 1.

### 3. PRACTICE

The primary difficulty in reaching large values of  $n$  with our first algorithm is the amount of space it requires. One way to address this is to create a larger wheel, sieve more but shorter arithmetic progressions for primes, and

rely less on sieving and more on primality tests (in the style of [28]) when searching for  $k$ -tuples.

We use the sieve of Eratosthenes instead of the Atkin-Bernstein sieve for the arithmetic progressions, and this is the source of the  $\log \log n$  factor slowdown. The gain here is less space needed by a factor of  $k$ , and the effective trial division performs a quantifiable filtering out of non-primes.

Instead of sieving by all primes up to  $\sqrt{n}$ , we sieve only by primes up to a bound  $B := \exp O(\log n / \log \log n)$ . In practice, we choose  $B$  so everything fits in CPU cache. We then use a base-2 pseudoprime test, followed by a prime test. For smaller  $k$ , we use the pseudosquares prime test of Lukes, Patterson, and Williams [20], which is fast and deterministic, assuming a sufficient table of pseudosquares is available, and importantly it takes advantage of the trial division effect of the sieve of Eratosthenes. For larger  $k$ , we can simply use the AKS prime test [1].

This change means we can get by with only  $O(B)$  space. Choosing  $B$  larger or smaller through its implied constant makes a tradeoff between the cost of sieving and the cost of performing base-2 pseudoprime tests.

### 3.1. Our Second Algorithm.

- (1) Choose  $B := \exp O(\log n / \log \log n)$  (this can be  $2^{\lfloor c \log n / \log \log n \rfloor}$  for a small positive constant  $c$ ). We begin by finding the list of primes up to  $B$  and dividing them into the two sets  $\mathcal{W}$  and  $\mathcal{S}$ . Small primes go into  $\mathcal{W}$  and the remainder go in  $\mathcal{S}$ . We want  $W := \prod_{p \in \mathcal{W}} p$  to be as large as possible with the constraint that  $W \leq n/B$ . This implies that the largest prime in  $\mathcal{W}$  will be roughly  $\log n$  in size.
- (2) If  $k \leq 6$ , we will need to perform the pseudosquares prime test, so in preparation, find all pseudosquares  $L_p \leq n/B$  (see [20]).
- (3) Next, as before, we construct the wheel data structure so that it will generate all possible correct residues modulo  $W$ .
- (4) For each residue  $r \bmod W$  generated by the wheel, we construct a bit vector  $\mathbf{v}[]$  of length  $n/W$ . Each vector position  $\mathbf{v}[j]$ , for  $j = 0, \dots, \lfloor n/W \rfloor$ , represents the  $x = x(j)$  value  $x(j) = r + j \cdot W$  for the  $k$ -tuple  $(f_1(x(j)), f_2(x(j)), \dots, f_k(x(j)))$ . We initialize  $\mathbf{v}[j]=1$ , but clear it to 0 if we find a prime  $p \in \mathcal{S}$  where  $p \mid f_i(x(j))$  for some  $i$ .

```

for( p ∈ S)
  winv=W-1 mod p;
  for(i=0; i<k; i++)
    j=winv*(-biai-1-r) mod p;
    while(j<n/W)
      v[j]=0;
      j=j+p;

```

Once this sieving is complete, the only integers  $j$  with  $v[j] = 1$  that remain, satisfy the property that all the  $f_i(x(j))$  have no prime divisors less than  $B$ .

- (5) For each such  $x(j)$  remaining (that is,  $v[j] = 1$ ), we first do a base-2 strong pseudoprime test on  $f_1(x(j))$ . If it fails, we cross it off (set  $v[j] = 0$ ). If it passes, we try  $f_2(x)$  and so forth, keeping  $v[j] = 1$  only if all  $k$  values  $f_i(x(j))$  pass the pseudoprime test. We then perform a full prime test on the  $f_i(x(j))$  for all  $i$ . If  $k \leq 6$ , we use the Lukes, Patterson, and Williams pseudosquares prime test [20] as done in [28]. For larger  $k$ , we use the AKS prime test [1]. (This is for the purposes of the theorem; in practice, the pseudosquares prime test is faster, so we use that instead.) If all the  $f_i(x(j))$  pass the prime tests, the corresponding  $k$ -tuple is written for output.

In practice, this version of the algorithm works best for  $k \geq 4$ . For very small  $k$ , the prime tests become the runtime bottleneck, and so we recommend using  $B = \sqrt{n}$  so that the base-2 pseudoprime tests and the pseudosquares prime test are not needed, as the sieving will leave only primes.

**3.2. Complexity Analysis.** Finding the primes up to  $B$  takes at most  $O(B)$  time using  $O(\sqrt{B})$  space, well within our bounds. See [28] for a sublinear time algorithm to find all needed pseudosquares. In practice, all pseudosquares up to  $10^{25}$  are known [29]. The cost in time and space to build the wheel is, up to a constant factor, the same. So we now focus on steps (4) and (5).

As shown above, the number of residues to check mod  $W$  is

$$\prod_{p \in \mathcal{W}} (p - k) = W \prod_{p \in \mathcal{W}} \frac{p - k}{p} \ll W \left( \frac{e^{-\gamma}}{\log \log n} \right)^k.$$

The time to sieve each interval of length  $n/W$  using primes up to  $B$  is at most proportional to

$$\sum_{p \leq B} \frac{kn}{pW} \ll \frac{kn \log \log B}{W} \sim \frac{kn \log \log n}{W}.$$

Here the multiplier  $k$  is required because we cross off  $k$  residues modulo all but finitely many primes  $p \leq B$ .

By Mertens's theorem, at this point an average of at most

$$\frac{n}{W} \sum_{p \leq B} \left( 1 - \frac{1}{p} \right) \ll \frac{n}{W \log B} \ll \frac{n \log \log n}{W \log n}$$

vector locations remain to be prime tested. (Note that we cannot make any assumptions about the relative independence of the primality of the  $f_i(x)$  values for different  $i$ , and so we cannot use a  $(1 - k/p)$  factor here.) A single base-2 strong pseudoprime test takes at most  $O(\log n)$  operations to



perform, for a total cost proportional to

$$\frac{kn \log \log n}{W \log n} \log n \sim \frac{kn \log \log n}{W}$$

arithmetic operations to do the base-2 strong pseudoprime tests for each value of  $r \bmod W$ . This matches the sieving cost of  $O(kn \log \log n/W)$  from above. (Note that if we deliberately choose a larger value for  $B$ , the increased sieving will decrease the number of pseudoprime tests needed. This tradeoff can be used to fine-tune the running time of the algorithm.)

Thus, the total cost for sieving and base-2 pseudoprime tests is

$$O\left(\frac{kn}{(\log \log n)^{k-1}}\right),$$

which we obtain by multiplying by the number of residues  $O(W/(\log \log n)^k)$ .

Next we need to count integers that pass the base-2 strong pseudoprime test. Such integers are either prime, or composite base-2 pseudoprimes. We switch to counting across all residues  $r \bmod W$  to obtain an overall bound.

Lemma 1 tells us that at most  $O(n/(\log n)^k)$  integers are prime that fit the pattern, so this is an upper bound on primes that pass the base-2 pseudoprime test.

Pomerance [23] showed that the number of composite base-2 pseudoprimes is bounded by

$$ne^{-\sqrt{\frac{\log n \log \log \log n}{\log \log n}}} \ll \frac{n}{(\log n)^{k+1}}$$

which is negligible. This plus the bound for primes above gives us the  $O(n/(\log n)^k)$  bound we desire for all integers that pass the base-2 pseudoprime test.

Next, to bound the cost of prime tests, we have two cases:  $k > 6$ , or  $2 < k < 6$ .

For  $k > 6$ , we use the AKS prime test [1, 18] which takes time  $O((\log n)^{6+o(1)})$ . The cost of applying the AKS prime test to all the integers  $f_i(x)$  after they all pass a base-2 pseudoprime test is at most proportional to

$$k \cdot (\log n)^{6+o(1)} \cdot \frac{n}{(\log n)^k} \ll \frac{kn}{(\log n)^{k-6+o(1)}}$$

which is  $o(kn/(\log \log n)^k)$  for  $k > 6$ .

Note that when  $k$  is large, in practice we might only do the base-2 pseudoprime tests, and then run full prime tests on the output afterwards, since the amount of output will be rather small.

For  $2 < k \leq 6$ , Conjecture 1 implies that the pseudosquares prime test takes  $O((\log n)^2)$  arithmetic operations to test integers  $\leq n$  for primality, given a table of pseudosquares  $\leq n$ . If  $n$  has no prime divisors below  $B$ , then pseudosquares up to  $n/B$  suffice. See [20, 28].

So, under the assumption of Conjecture 1, the cost of applying the pseudosquares prime test to all the integers  $f_i(x)$  after they all pass a base-2

pseudoprime test is at most proportional to

$$k \cdot (\log n)^2 \cdot \frac{n}{(\log n)^k} \ll \frac{kn}{(\log n)^{k-2}}$$

and this is  $o(kn/(\log \log n)^k)$  for  $k > 2$ .

The space used is dominated by the length of the sieve intervals and the space needed to store the primes in  $\mathcal{S}$ , which is  $O(B)$  bits.

This completes the proof of Theorems 2 and 3.

#### 4. COMPUTATIONS

As mentioned previously, we implemented several versions of our second algorithm to see what we could compute. We looked for new computational records that were within reach of our university's nice but aging hardware. Below we discuss some of the results of those computations. Some of the implementation details are specific to a particular computation. Here are a few remarks about implementation details that these computations had in common.

- We wrote our programs in C++ using the Gnu compiler under linux. GMP was used for multiprecision arithmetic when necessary. Note that it is fairly easy to write the code such that GMP was needed only on occasion and for prime tests.
- We used MPI and ran our code on Butler University's cluster *Big Dawg*. This machine has 16 compute nodes with 12 cores (2 CPUS) each at optimal capacity; our average utilization was around 150 of the 192 cores due to compute nodes going down from time to time. The CPU is the Intel Xeon CPU E5-2630 0 @ 2.30GHz with 15 MB cache, with 6 cores per CPU.

To parallelize the algorithm, we striped on the residues  $r \bmod W$ ; all processes stepped through all the  $r \bmod W$  residues, but only sieved for primes for their chosen residues. This meant there was very little communication overhead except for when periodic checkpoints were done, about every 15-30 minutes.

- We usually chose our wheel size ( $W$ ) and sieve intervals so that the size of each interval ( $n/W \approx B$ ) was at most a few megabytes so that it would fit in the CPU cache. We used a `vector<bool>`, which packs bits.
- For larger values of  $k$ , we observed that when sieving by smaller primes  $p$  by each of the  $k$  residues, we might find that almost all the bits of the current interval were cleared long before we reached the sieving limit  $B$ , so we created a simple early-abort strategy that was able to save time.

The very few remaining bits were tested with the base-2 strong pseudoprime test even though we had not sieved all the way to  $B$ . We also, then, replaced the use of the pseudosquares prime test with

the Miller-Rabin test [21, 24] with the results from [27], due to the spotty trial-division information.

**4.1. Twin Primes and Brun’s Constant.** Let  $\pi_2(X)$  count the twin prime pairs  $(p, p + 2)$  with  $p < X$  and  $S_2(X)$  be the sum of the reciprocals of their elements. Thomas Nicely computed these functions up to  $2 \cdot 10^{16}$  (See <http://www.trnicely.net/#PI2X>). We verified his computations and extended the results to  $X = 10^{17}$ . A portion of our computational results are in the table below.

$X$	$\pi_2(x)$	$S_2(X)$
$1 \cdot 10^{16}$	10304195697298	1.83048442465833932906
$2 \cdot 10^{16}$	19831847025792	1.83180806343237985727
$3 \cdot 10^{16}$	29096690339843	1.83255992186282759050
$4 \cdot 10^{16}$	38196843833352	1.83308370147757159450
$5 \cdot 10^{16}$	47177404870103	1.83348457901336613822
$6 \cdot 10^{16}$	56064358236032	1.83380868220200440399
$7 \cdot 10^{16}$	64874581322443	1.83408033035537994465
$8 \cdot 10^{16}$	73619911145552	1.83431390342560497644
$9 \cdot 10^{16}$	82309090712061	1.83451860315233433306
$10 \cdot 10^{16}$	90948839353159	1.83470066944140434160

The last section of Klyve’s PhD Thesis [17] describes how to use this information to derive bounds for Brun’s constant.

We have a few remarks on our algorithm implementation:

- As mentioned above, for small  $k$  like  $k = 2$ , it is more efficient to set  $B = \sqrt{n}$  so that sieving also determines primality, thereby avoiding base-2 strong pseudoprime tests and primality tests.
- We computed  $S_2$  using Kahan summation [16] with the `long double` data type in C++, which gave us 17 digits of accuracy; Thomas Nicely has data with 53 digits of accuracy. The partial sums were accumulated in 10,000 buckets for each process, and then the buckets were in turn added up across processes using Kahan summation.
- Our computation took roughly 3 weeks of wall time, which included at least one restart from a checkpoint. Our verification of Nicely’s work to  $10^{16}$  took 42 hours.
- We used a wheel with  $W = 6469693230$ . Note that this is roughly  $20 \cdot \sqrt{10^{17}}$ . There were 214708725 residues  $r \bmod W$  to sieve.

See [OEIS.org](http://oeis.org) sequence A007508.

**4.2. Quadruple Primes.** A related sum involves the reciprocals of the elements of the prime tuple  $(p, p + 2, p + 6, p + 8)$ . Let  $\pi_4(X)$  count these tuples up to  $X$ , and let  $S_4(X)$  be the sum of the reciprocals of their elements. Thomas Nicely computed these functions up to  $2 \cdot 10^{16}$ . We extended this computation and partial results are in the table below. The first two lines are Thomas Nicely’s own results, which we verified.

$X$	$\pi_4(x)$	$S_4(X)$
$1 \cdot 10^{16}$	25379433651	0.87047769123404594005
$2 \cdot 10^{16}$	46998268431	0.87048371094805250092
$3 \cdot 10^{16}$	67439513530	0.87048703104321483993
$4 \cdot 10^{16}$	87160212807	0.87048930200258802756
$5 \cdot 10^{16}$	106365371168	0.87049101694672496876
$6 \cdot 10^{16}$	125172360474	0.87049238890880442047
$7 \cdot 10^{16}$	143655957845	0.87049352884516002359
$8 \cdot 10^{16}$	161868188061	0.87049450175556017194
$9 \cdot 10^{16}$	179847459283	0.87049534891720052192
$10 \cdot 10^{16}$	197622677481	0.87049609811047504740

This computation took about 4 days, and we used a separate program rather than looking for pairs of twin primes in the first program. Even though  $k = 4$  is large enough to use prime tests, we found that sieving to  $\sqrt{n}$  was faster in practice.

We used a wheel with  $W = 200560490130$  which gave 472665375 residues. See [OEIS.org](https://oeis.org/A050258) sequence A050258.

**4.3. Cunningham Chains.** We have two computational results for Cunningham chains.

- (1) We found the smallest chain of length 15 of the first kind, and it begins with the prime

$$p = 90616\ 21195\ 84658\ 42219.$$

The next few chains of this length of the first kind are

$$1\ 13220\ 80067\ 50697\ 84839$$

$$1\ 13710\ 75635\ 40868\ 11919$$

$$1\ 23068\ 71734\ 48294\ 53339$$

$$1\ 40044\ 19781\ 72085\ 69169$$

This computation took roughly a month of wall time. Here we used wheel size  $W = 19835154277048110$ , with 12841500672 residues to sieve.

See [OEIS.org](https://oeis.org/A005602) sequence A005602.

- (2) In 2008 Jaroslaw Wroblewski found a Cunningham chain of length 17 of the first kind, starting with

$$p = 27\ 59832\ 93417\ 13865\ 93519,$$

and we were able to show that this is in fact the smallest such chain of that length.

This computation took roughly three months of wall time. We used  $W = 1051263176683549830$  with 35864945424 residues to sieve. With roughly three times as many residues as the previous computation, it took roughly three times as long to complete.

## 5. DISCUSSION AND FUTURE WORK

In summary, we have described and analyzed two algorithms for finding primes in patterns, and then shown that the second of these algorithms is quite practical by performing a few computations.

We have some ideas for future work.

- In the Introduction, we mentioned that our algorithms could be used to find Charmichael numbers by finding prime triplets that satisfy the pattern  $(6x + 1, 12x + 1, 18x + 1)$ , but we have not yet done that computation [6].
- Does it make sense to use Bernstein's doubly-focused enumeration to attempt to further reduce the running time? See [5, 29, 30]
- A natural extension to our algorithms here is to allow the linear polynomials  $f_i$  to potentially be higher degree, irreducible polynomials. See Schinzel's Hypothesis H (See [26] and [7, §1.2.2]) and the Bateman-Horn conjecture [4].

## ACKNOWLEDGEMENTS

We wish to thank Frank Levinson for his generous gift that funds the Butler cluster supercomputer *Big Dawg*, and the Holcomb Awards Committee for their financial support of this work.

We also wish to thank an anonymous referee for many detailed, helpful comments on an earlier version of this paper.

## REFERENCES

- [1] Manindra Agrawal, Neeraj Kayal, and Nitin Saxena. PRIMES is in P. *Ann. of Math. (2)*, 160(2):781–793, 2004.
- [2] A. O. L. Atkin and D. J. Bernstein. Prime sieves using binary quadratic forms. *Mathematics of Computation*, 73:1023–1030, 2004.
- [3] Eric Bach and Lorenz Huelsbergen. Statistical evidence for small generating sets. *Math. Comp.*, 61(203):69–82, 1993.
- [4] Paul T. Bateman and Roger A. Horn. A heuristic asymptotic formula concerning the distribution of prime numbers. *Math. Comp.*, 16:363–367, 1962.
- [5] Daniel J. Bernstein. Doubly focused enumeration of locally square polynomial values. In *High primes and misdemeanours: lectures in honour of the 60th birthday of Hugh Cowie Williams*, volume 41 of *Fields Inst. Commun.*, pages 69–76. Amer. Math. Soc., Providence, RI, 2004.
- [6] Jack Chernick. On Fermat's simple theorem. *Bull. Amer. Math. Soc.*, 45(4):269–274, 1939.
- [7] R. Crandall and C. Pomerance. *Prime Numbers, a Computational Perspective*. Springer, 2001.
- [8] L. E. Dickson. A new extension of Dirichlet's theorem on prime numbers. *Messenger of mathematics*, 33:155–161, 1904.
- [9] Tony Forbes. Prime clusters and Cunningham chains. *Math. Comp.*, 68(228):1739–1747, 1999.
- [10] William F. Galway. Dissecting a sieve to cut its need for space. In *Algorithmic number theory (Leiden, 2000)*, volume 1838 of *Lecture Notes in Comput. Sci.*, pages 297–312. Springer, Berlin, 2000.

- [11] Daniel M. Gordon and Gene Rodemich. Dense admissible sets. In *Algorithmic number theory (Portland, OR, 1998)*, volume 1423 of *Lecture Notes in Comput. Sci.*, pages 216–225. Springer, Berlin, 1998.
- [12] Richard K. Guy. *Unsolved problems in number theory*. Problem Books in Mathematics. Springer-Verlag, New York, third edition, 2004.
- [13] H. Halberstam and H.-E. Richert. *Sieve Methods*. Academic Press, 1974.
- [14] G. H. Hardy and J. E. Littlewood. Some problems of ‘partitio numerorum’; iii: On the expression of a number as a sum of primes. *Acta Mathematica*, 44(1):1–70, Dec 1923.
- [15] G. H. Hardy and E. M. Wright. *An Introduction to the Theory of Numbers*. Oxford University Press, 5th edition, 1979.
- [16] W. Kahan. Pracniques: Further remarks on reducing truncation errors. *Commun. ACM*, 8(1):40–, January 1965.
- [17] Dominic Klyve. *Explicit Bounds on Twin Primes and Brun’s Constant*. PhD in mathematics, Dartmouth College, Hanover, NH USA, 2007.
- [18] H. W. Lenstra, Jr. Primality testing with gaussian periods. In *Proceedings of the 22Nd Conference Kanpur on Foundations of Software Technology and Theoretical Computer Science*, FST TCS ’02, pages 1–, Berlin, Heidelberg, 2002. Springer-Verlag.
- [19] Günter Löh. Long chains of nearly doubled primes. *Math. Comp.*, 53(188):751–759, 1989.
- [20] R. F. Lukes, C. D. Patterson, and H. C. Williams. Some results on pseudosquares. *Math. Comp.*, 65(213):361–372, S25–S27, 1996.
- [21] G. Miller. Riemann’s hypothesis and tests for primality. *Journal of Computer and System Sciences*, 13:300–317, 1976.
- [22] T. Nicely. Enumeration to  $10^{14}$  of the twin primes and Brun’s constant. *Virginia J. Sci.*, 46:195–204, 1996.
- [23] Carl Pomerance. On the distribution of pseudoprimes. *Math. Comp.*, 37(156):587–593, 1981.
- [24] M. O. Rabin. Probabilistic algorithm for testing primality. *Journal of Number Theory*, 12:128–138, 1980.
- [25] Hans Riesel. *Prime Numbers and Computer Methods for Factorization*, volume 126 of *Progress in Mathematics*. Birkhäuser, 2nd edition, 1994.
- [26] A. Schinzel and W. Sierpiński. Sur certaines hypothèses concernant les nombres premiers. *Acta Arith.*, 4:185–208; erratum 5 (1958), 259, 1958.
- [27] Jonathan Sorenson and Jonathan Webster. Strong pseudoprimes to twelve prime bases. *Math. Comp.*, 86(304):985–1003, 2017.
- [28] Jonathan P. Sorenson. The pseudosquares prime sieve. In Florian Hess, Sebastian Pauli, and Michael Pohst, editors, *Proceedings of the 7th International Symposium on Algorithmic Number Theory (ANTS-VII)*, pages 193–207, Berlin, Germany, July 2006. Springer. LNCS 4076, ISBN 3-540-36075-1.
- [29] Jonathan P. Sorenson. Sieving for pseudosquares and pseudocubes in parallel using doubly-focused enumeration and wheel datastructures. In Guillaume Hanrot, Francois Morain, and Emmanuel Thomé, editors, *Proceedings of the 9th International Symposium on Algorithmic Number Theory (ANTS-IX)*, pages 331–339, Nancy, France, July 2010. Springer. LNCS 6197, ISBN 978-3-642-14517-9.
- [30] Kjell Wooding and H. C. Williams. Doubly-focused enumeration of pseudosquares and pseudocubes. In *Proceedings of the 7th International Algorithmic Number Theory Symposium (ANTS VII)*, Berlin, Germany, 2006.
- [31] Yitang Zhang. Bounded gaps between primes. *Ann. of Math. (2)*, 179(3):1121–1174, 2014.

COMPUTER SCIENCE AND SOFTWARE ENGINEERING, BUTLER UNIVERSITY, INDIANAPOLIS, IN 46208 USA

*E-mail address:* `sorenson@butler.edu`

MATHEMATICS, STATISTICS, AND ACTUARIAL SCIENCE, BUTLER UNIVERSITY, INDIANAPOLIS, IN 46208 USA

*E-mail address:* `jwebste@butler.edu`