k-Parent Aliquot Numbers: Interm Report Concerning the Image of the Sum-of-Proper-Divisors Function

Gavin Guinn
Supervisor: Michael J. Jacobson

December 2021

1 Introduction

If we examine numbers in the set $\{6, 28, 496, 8128, ...\}$ we notice that every member n is equal to the sum of its divisors with the exclusion of itself,

$$1 + 2 + 3 = 6.$$

This of course is the definition of a *perfect number*, a property that has fascinated number theorists since antiquity. The *sum-of-divisors* function is defined for $n > 1$ by

$$\sigma(n) = \sum_{d|n} d.$$

The *proper-divisors* of n refers to the subset of the divisors which excludes itself, the *sum-of-proper-divisors* is defined for $n > 1$ by

$$s(n) = \sigma(n) - n,$$

thus n is a perfect number if and only if $s(n) = n$. The sum-of-proper-divisors is a natural way to consider perfect numbers but is certainly not limited to that domain. For instance consider $s(9) = 4$ which has the property $n < s(n)$, such numbers are *deficient* while numbers that satisfy $n > s(n)$ are *abundant*.

Aliquot sequences are produced by iterating $s(n)$, for example consider the

sequence starting at 12:

$$\begin{aligned} s(12) &= 1+2+3+4+6 \\ s(16) &= 1+2+4+8 \\ s(15) &= 1+3+5 \\ s(9) &= 1+3 \\ s(4) &= 1+2 \\ s(3) &= 1 \end{aligned}$$

In this case the sequence has terminated as $s(n)$ is only defined for $n > 1$, this sequence is also bounded as no value is greater than 16. A sequence does not have to terminate to be bounded, since perfect numbers are defined by $s(n) = n$ an aliquot sequence that reaches a perfect number will cycle. The members of a cycle of length 2 are referred to as an *amicable pair*, for instance

$$s(1184) \to s(1210) \to s(1184) \to ...$$

It seems that cycles can form with arbitrary length [**?**] note a cycle of length 24, numbers in such cycles are *sociable*. Aliquot sequences are sometimes said to terminate in a cycle, we will only refer to aliquot sequences that reach 1 to have terminated while referring to the latter case as a bounded sequence. Aliquot sequences are formalized by [**?**] in the following way:

Definition 1.1. The aliquot sequence starting at $a_0 > 1$ is defined by

$$a_0,\ a_1 = s(a_0),\ a_2 = s^2(a_0),\ ...$$

Proposition 1.1. An aliquot sequence is bounded if and only if

$$\exists a_n,\ \forall a_m \text{ s.t. } a_n \geq a_m$$

The aliquot sequence of 276,

$$s(276) \to s(396) \to s(696) \to s(1104) \to s(1872) \to ...$$

is the smallest example that has not been shown to be bounded, [**?**] has followed the it's growth over 2090 iterations to values with over 200 digits! The dichotomy between the seemingly unbounded growth of the 276 sequence and the boundedness of the 12 sequence gets to the most important open question relating to aliquot sequences:

Conjecture 1.1 (Catalan-Dickson)**.** All aliquot sequences are bounded and thus must terminate or enter a cycle. [**?**]

Conjecture 1.2 (Guy-Selfridge)**.** The Catalan-Dickson conjecture is false, perhaps most aliquot sequences are unbounded. [**?**]

The computational exploration of these sequences that has taken place since Catalan originally proposed his conjecture in 1888 has certainly given credence to the Guy-Selfridge point-of-view. However a trend in the data is far from a proof, under our current understanding it is entirely possible that the 276 sequence could trip over a prime or enter a cycle on some future iteration.

2 Motivation

What statistics can be compiled on the behaviour of $s(n)$ to argue for or against the Calatan-Dickson Conjecture? The authors of [**?**] consider numerical and analytical approaches to determine the average of $s(n)/n$. The average order of a arithmetic function is a simple function that has the same behaviour on average as the less tractable function. Define $\mathcal{S}(n)$ to be the average order of $s(n)$, when computing the average of $s(n)/n$ we are numerically estimating $\mathcal{S}(n)$. When considering this metric [**?**] restrict the domain to evens to examine $\mathcal{S}(2n)$. I believe the impetus for this decision can be traced to [**?**] who heuristically argue that unbounded sequences exist:

- **if** $\mathcal{S}(2n) > 2n$ and,
- **if** other effects on the growth of a sequence are relatively minor compared to the tendency for aliquot sequences to preserve parity,
- **then** we would expect most even aliquot sequences that reach large values to be unbounded.

This argument seems to match the numerical evidence as seemly unbounded sequences such as $s(276)$ [**?**] appear to maintain constant even parity.

The method used to either analytically determine or numerically compute $\mathcal{S}(2n)$ impacts the parameters value. While not explicitly stated by Guy and Selfridge, [**?**] indicate that arithmetic mean of $s(2n)/2n$ was used to determine $\mathcal{S}(2n)$ in constructing their heuristic argument for unboundedness.

Theorem 2.1. Arithmetic Mean of $s(2n)/2n$

$$\lim_{N\to\infty} \frac{1}{N} \sum_{n=1}^{N} \frac{s(2n)}{2n} = \left(\frac{5\pi^2}{24} - 1 \right) = 1.0562...$$

With this metric we have $\mathcal{S}(2n) = (2n)(1.0563...)$ and thus $\mathcal{S}(2n) > 2n$. However as pointed out by [**?**] the sequence $\{5, \frac{1}{5}, 5, \frac{1}{5}, ...\}$ is bounded but its arithmetic mean tends to 2.6, a better metric for evaluating the boundedness of a sequence is to consider the geometric mean.

Theorem 2.2. Geometric Mean of $s(2n)/2n$

$$\mu = \lim_{N\to\infty} \exp\left(\frac{1}{N} \sum_{n=1}^{N} \log s(2n)/2n \right) = 0.969...$$

With this more accurate metric we have $\mathcal{S}(2n) < 2n$ which suggests that even aliquot sequences will in fact decrease on average! This result certainly throws cold water on the heuristic argument of Guy and Selfridge. However the monumental growth of sequences like $s(276)$ suggests that this metric does not completely describe sequence behaviour.

To improve this model of sequence behaviour the authors of [**?**] consider the image of $s(2n)$ when computing $\mathcal{S}(2n)$. Consider the solutions to $s^{-1}(316)$,

$$s(192) = s(304) = s(344) = s(412) = 316,$$

since multiple sequences are tributary at 316 it seems reasonable to conclude that the value of $s(316)/316$ impacts average sequence growth more than numbers with fewer pre-images. [**?**] explores this by constructing an average of $s(2n)/2n$ weighted by $\#s^{-1}(2n)$. Counting the number of pre-images under $s(2n)$ was certainly on the mind of Dr. Guy when he posed the question that motivates this project:

> Think of a number!! Say 36%, which is nice and divisible. It appears that about 36% of the even numbers are "orphans".
>
> Divide by 1. For about 36% of the (even) values of n there is just one positive integer m such that $s(m) = n$. These values of n have just one "parent".
>
> Divide by 2. About 18% of the even values of n have exactly two parents.
>
> Divide by 3. About 6% of the even values of n have three parents.
>
> Divide by 4. About 1.5% of the even values of n have just 4 parents.
>
> This suggests that $1/(e \cdot p!)$ of the even numbers have p parents.
>
> Experiments suggests that these values are a bit large for small values of p and a bit small for larger values of p. Can anything be proved?

Definition 2.1. n is a k-parent aliquot number if $s^{-1}(n)$ has k solutions.

These percentages refer to the density of even k-parent numbers or the probability of encountering a k-parent number when choosing randomly from all even natural numbers. There are a suite of colourful names to refer to numbers outside of the range of $s(\cdot)$ including *non-aliquots* and *untouchable numbers*, for the sake of consistency I will be referring to all such numbers as *non-aliquot*.

3 Project

3.1 Project Preamble

It is worthwhile to first outline some subtlety around what we are counting when considering k-parent numbers and how those counts relate to the existing literature. An important observation when counting non-aliquots is that all odd numbers are in the image of $s(\cdot)$ (with the exception of 5) [**?**], so we only need to consider even numbers when counting non-aliquots. A distinction is that that non-aliquots are numbers outside the image of $s(\cdot)$ while k-parent numbers consider how the image of $s(\cdot)$ is distributed.

In posing his question Dr. Guy considers the density of even k-parent numbers over the even numbers.[1][2] This approach differs from that used in the literature around non-aliquots; [**?**], [**?**], and [**?**] instead consider the density of non-aliquots over all natural numbers. Thus to match the existing literature we will consider the density of *even* k-parent numbers over *all natural numbers*. This is slightly awkward as odd k-parent numbers exist but we are only considering the density of even k-parent numbers over both the evens and odds. However we can take some solace in the fact that densities generated with this method can doubled to convert back to Dr. Guy's approach. A special case to consider is the odd non-aliquot 5 which will be excluded from the count of even 0-parent numbers, this will result in such counts being exactly one less than those reported for non-aliquots. With these consideration outlined we can now move to the project particulars.

3.2 Previous Work

It is useful to observe that k-parent aliquot numbers are a generalization of non-aliquots. Thus an effective strategy for quantifying the density of k-parent numbers is to generalize techniques such as the heuristic approach of [**?**]. The authors employ statistical methods to model the density of non-aliquots, this model closely matches experimental observations.

Conjecture 3.1. Heuristic Model for the Density of Non-Aliquots

$$\Delta = \lim_{y\to\infty} \frac{1}{\log y} \sum_{\substack{a\leq y \\ 2|a}} \frac{1}{a} \mathrm{e}^{-a/s(a)}$$

During a previous work term I produced the following tentative generalization of this model.

[1] This choice may have been made to relate this result back to the Guy-Selfridge heuristic argument for the asymptotic unboundedness of even sequences

[2] While it is not the subject of this project it would be perfectly reasonable to consider the the densities of all k-parent numbers

Conjecture 3.2. Heuristic Model for the Density of Even k-Parent Numbers

$$\Delta_k = \lim_{y\to\infty} \frac{1}{\log y} \sum_{\substack{a\leq y \\ 2|a}} \frac{a^{k-1}}{k!\cdot s(a)^k} \cdot \mathrm{e}^{-a/s(a)}$$

A priority for this project is to fully justify this result. Some values are given in Table 1.

Δ_k	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
$y=10^3$	0.155	0.164	0.100	0.046	0.018
$y=10^4$	0.161	0.165	0.098	0.044	0.017
$y=10^5$	0.165	0.166	0.097	0.044	0.016
$y=10^6$	0.167	0.166	0.096	0.043	0.016
$y=10^7$	0.169	0.167	0.096	0.042	0.016
$1/(p!\cdot \mathrm{e})$	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
-	0.378	0.368	0.184	0.061	0.015
$1/2(p!\cdot \mathrm{e})$	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
-	0.184	0.184	0.092	0.031	0.008

Table 1: Approximation of Δ_k compared to Dr. Guy's estimates

A complementary approach is to directly compute the densities of k-parent numbers. If the computed densities closely match the heuristic model we would have strong evidence for correctness. [**?**] leverage an algorithm of [**?**] to enumerate the image of $s(2n)$ within a bounding range, the percentage of numbers in this range that are not enumerated is the density of non-aliquots. During that previous work term I successfully generalized this technique to compute the densities of k-parent numbers, however this code is currently bugged such that the counts of 0-parent numbers diverge from the published counts of non-aliquots at the bound of 2^{40}. Densities computed by this program are compared to values of Δ_k and Dr. Guys estimates in Table 2.

$a_k(n)/n$	$k=0$	$k=1$	$k=2$	$k=3$	$k=4$
$n=10^3$	0.088	0.202	0.148	0.048	0.009
$n=10^4$	0.121	0.187	0.121	0.048	0.015
$n=10^5$	0.139	0.182	0.108	0.045	0.016
$n=10^6$	0.150	0.178	0.103	0.042	0.016
$n=10^7$	0.157	0.175	0.099	0.042	0.016
$n=10^8$	0.162	0.173	0.097	0.041	0.016
$n=10^9$	0.166	0.171	0.096	0.040	0.015

y	Δ_0	Δ_1	Δ_2	Δ_3	Δ_4
$y=10^7$	0.169	0.167	0.096	0.042	0.016

$1/2(p!\cdot \mathrm{e})$	$p=0$	$p=1$	$p=2$	$p=3$	$p=4$
-	0.184	0.184	0.092	0.031	0.008

Table 2: $a_k(n)$ is a counting function for k-parent aliquot numbers less than or equal to n, $a_k(n) = |\{x \mid \#s(x)^{-1} = k,\ 1 \leq x \leq n\}|$.

3.3 Project Goals

The goals for this project are:

1. Correct my previous implementation of the Pomerance-Yang (P-Y) algorithm [**?**] that produced a off-by-one error at bound 2^{40}. Steps in this debugging process include:

   (a) Ensure correctness of $\sigma(\cdot)$ enumeration which is required by the P-Y algorithm. I will compare the outputs of this implementation with other algorithms for enumerating $\sigma(\cdot)$ up-to 2^{40}. If multiple distinct algorithms produce identical output we will have reasonably strong evidence for the enumerations correctness.

   (b) If the above method does not identify the source if the bug I will compare the intermediate results of my implementation of P-Y with the implementation used in [**?**]. This could be done by running the implementations in parallel and comparing the characteristic functions after each block is completed. This computationally and memory expensive method would certainly identify the value in which the implementations differ.

2. Formalize the argument justifying my generalization of the heuristic model of [**?**] for approximating the density of non-aliquot numbers. It remains to ensure correctness of my argument and rewrite in a formal style.

3. A related goal to correcting my P-Y implementation of is to tune the inputs to ensure optimal performance. For instance setting the length to be sieved over in a step incorrectly can result in an order-of-magnitude increase in run-time. Choosing optimized values for all these constants in

a multi-threaded context is non-trivial and would likely involve tuning to the specifications of the hardware running the algorithm (i.e. CPU cache).

4. A secondary goal to correcting my implementation of P-Y is to research alternative algorithms for enumerating the sum-of-proper divisors. This algorithm dominates the run-time of P-Y so improvements are highly desirable.

5. A small side goal is to recompute the weighted geometric mean of $s(n)/n$ weighted by 1 if n is an aliquot orphan and $\#s^{-1}(n)$ otherwise. The rational for this is explained in subsection 5.3.

6. If time allows I will seek to generalize the provable lower bounds of [**?**] for the density of non-aliquot numbers. I would generalize this work to prove lower bounds for the densities of k-parent numbers.

The final goal of the project is to produce a paper that presents my heuristic model for k-parent numbers and utilizes the computed densities of k-parent as evidence for that result. The correctness of computational work will be thoroughly tested and the code will be available in a well-documented GitHub repository. This work will be motivated in the context of Dr. Guy's original problem statement and the Guy-Selfridge conjecture in general.

3.4 Project Status

The bulk of my time on this project has been spent reviewing the related literature, this context will certainly be useful when preparing the final report. I have also spent a good deal of time developing an algorithm for enumerating ranges of $\sigma(\cdot)$ similar to that of [**?**], a discussion these algorithms and the relationship to P-Y can be found in section 4. This algorithm could be useful in checking the correctness of the implementation I inherited from another researcher. This effort has given me experience working with sieving algorithms and insight into the challenges of tuning them for optimal performance.

I have also re-familiarized myself with my implementation of P-Y and attempted to re-run it on ARC, uCalgary's high performance cluster. The jobs I have submitted at bounds 10^{12} and 2^{40} have so far been unsuccessful due to either exceeding time or memory constraints, a priority is to reproduce the erroneous computation as a first step in debugging the error.

3.5 Project Timeline

I will to reproduce the erroneous computation as soon as possible; the delays associated with batch processing queues and the long computation time needed are the primary challenges to this deliverable. I will also confirm the correctness of the $\sigma(\cdot)$ enumeration as soon as possible, my work on an alternative $\sigma(\cdot)$

enumeration has already produced much of the computational machinery needed to do this. During the winter break I will take some time to compile all the disparate programming and writing I have produced so far into a single well-documented GitHub repository. Also during the break I will have a developed a method to tune this algorithm to ensure near optimal performance and memory usage.

Early in the winter semester, mid January 2022, I hope to have resolved the bug in my implementation of the P-Y algorithm. At a minimum by this point I will have compared my implementation of P-Y with that used in [**?**] and determined the value where the implementations vary. At this point I also will have computed the alternative geometric mean of $s(n)$ described above.

By the start of February 2022 I hope to have my work on the k-parent heuristic model summarized into a form appropriate for the final report. This will involve communicating with my supervisor to determine what details need to be included and what mathematical correctness issues need to be addressed.

Once these goals are achieved I will start writing my final paper and presentation which will summarize my effort on this project over both my NSERC work-term and the 502 course. If time allows I will explore either the provable lower bounds of [**?**] or faster algorithms for enumerating $\sigma(\cdot)$.

4 Discussion of Algorithms

To compute the density of k-parent aliquot numbers up-to bound X we must enumerate the image of all values n such that $s(n) \leq X$. For each $m \leq X$ we will have computed k distinct values of n such that $s(n) = m$, more concisely $s^{-1}(m) = \{n \,|\, s(n) = m\}$ and thus

$$\#s^{-1}(m) = |\{n \,|\, s(n) = m\}|.$$

$\#s^{-1}(m)$ is computed by looping through all values m corresponding to the previously enumerated n and iterating a counter in a array index corresponding to m.

The P-Y algorithm, described in subsection 5.1, is utilized to compute $s^{-1}(m)$. This algorithm requires $\sigma(n)$ for every odd value n between 1 and the upper bound X. Perhaps the simplest implementation is to directly compute these values of $\sigma(n)$ by directly factoring each value n using an approach like Pollard's Rho algorithm, however this approach is untenable at higher bounds. Sieving algorithms which produce enumerated ranges of $\sigma(\cdot)$ have a significant run-time advantage compared to enumerating a range with a one-at-a-time factoring algorithm.

4.1 Moews and Moews Algorithm for Enumerating $\sigma(\cdot)$

The current state-of-the-art for enumerating $\sigma(\cdot)$ is the segmented sieving algorithm of [**?**], Algorithm 1 is a transcription into pseudo-code. I believe this algorithm leverages the multiplicative property of $\sigma(\cdot)$ [**?**] to compute values as a product where p is prime and $n = p_1^{a_1} p_2^{a_2} \dots p_t^{a_t}$,

$$\sigma(n) = \sigma(p_1^{a_1})\sigma(p_2^{a_2}) \dots \sigma(p_t^{a_t}).$$

This strait-forward implementation of Moews and Moews (M-M) is inefficient, seemly due to the test on line 9 failing for most values of j. I inherited an optimized version of this algorithm that was used in [**?**] which seems to improve run-time shape; while I don't entirely understand the technique but it seems to reduce the number of tests by offsetting to values guaranteed to be divisible by p^e.

I cannot perform a direct performance comparison between the optimized and optimized implementations of M-M as I am not in possession of equivalent working implementations. However I do have working implementations of naive M-M and a inherited implementation of optimized M-M which enumerates $\sigma(n)$ for odd values of n; Table 3 presents a timing comparison to roughly demonstrate the need for optimization in making M-M performant.

Algorithm 1 M-M Algorithm to Enumerate Range of $\sigma(\cdot)$

```
1: procedure ENUMERATE SIGMA(N, M)                    ▷ N is length, M is base
2:     q[j] ← j + M, ∀j ∈ [0, N − 1]
3:     r[j] ← 1, ∀j ∈ [0, N − 1]
4:     for all primes p ≤ √(M + N − 1) do
5:         p^e ← p
6:         while p^e ≤ M + N do
7:             p^{e+1} ← p^e · p
8:             for all ∀j ∈ [0, N − 1] do
9:                 if p^e | (M + j) and p^{e+1} ∤ (M + j) then
10:                    q[j] ← q[j]/p^e
11:                    r[j] ← r[j] · ((p^{e+1} − 1)/(p − 1))        ▷ r[j] = r[j] · σ(p^e)
12:                end if
13:            end for
14:            p^e ← p^{e+1}
15:        end while
16:    end for
17:    for all ∀j ∈ [0, N − 1] do
18:        if q[j] ≠ 1 then
19:            r[j] ← r[j] · (q[j] + 1)
20:        end if
21:    end for
22:    return r
23: end procedure
```

-	Naive M-M	Optimized Odd M-M
$X = 10^4$	0.01s	0.00s
$X = 10^5$	0.07s	0.01s
$X = 10^6$	1.35s	0.04s
$X = 10^7$	32.69s	0.36s

Table 3: Timings comparison between naive M-M and optimized odd M-M, sieving $\sigma(\cdot)$ to bound X.

4.2 Segmented Accumulating Sieve for Enumerating $\sigma(\cdot)$

Before studying the M-M algorithm I implemented similar style of segmented sieve that computes enumerates $s(\cdot)$ from 1 to bound X such that the memory requirements are proportional to $\sqrt{X}$. This algorithm operates on a similar principle to the segmented sieve of Eratosthenes but instead of marking a number with a divisor non-prime it instead accumulates the divisor, I will refer to this as the segmented accumulating sieve (SAS) algorithm. The pseudo-code can be found in Algorithm 2.

Note that the SAS algorithm only enumerates $s(\cdot)$ between 1 and M while M-M can enumerate any selected range of $\sigma(\cdot)$. This is because the SAS algorithm relies on the state of buffers s and i to compute the next block. This is possibly correctable utilizing a trick to find the offset $m + X$ of a number n above a bound X,

$$m = (n - (X \bmod n)) \bmod n,$$

which I found in the optimized M-M implementation.

I cannot perform a direct performance comparison between the SAS algorithm and optimized implementations of M-M as I am not in possession of equivalent working implementations. However I can compare my SAS implementation which enumerates all $s(\cdot)$ up-to bound X with the optimized odd M-M which enumerates odd $\sigma(\cdot)$, this is presented in Table 4. Based on this limited experimental evidence it appears that both algorithms are linear to X with optimized odd M-M faster by a constant factor of 2; this constant run-time difference might be a result of optimized odd M-M sieving exactly half as many numbers as SAS.

An size of block being sieved over in a given step, B for Algorithm 2 and N for Algorithm 1, have dramatic performance implications. Timing and buffer use information for the SAS algorithm at various values of B are given in Table 5. The poor timings for 10^2 and 10^3 can likely be attributed to the algorithm spending to much time on loop controlling operations. The timings for 10^4 to 10^6 are all roughly similar, I hypothesize that this is because the buffers for these values sit comfortably in my CPU's 32 MiB L3 cache; I suspect the timings jump again at 10^7 because the buffer cannot all be cached simultaneously.

Algorithm 2 SAS Algorithm to Enumerate $s(\cdot)$

```
1: procedure ENUMERATE SN(M, B)                    ▷ M is max, B is block length
2:     s[i] ← 0, ∀i ∈ [0, B − 1]
3:     t[i] ← 1, ∀i ∈ [0, √M]                                  ▷ iterations buffer
4:     m[i] ← i, ∀i ∈ [0, √M]                                  ▷ multiple buffer
5:     for all i ∈ [B, 2B, ... , M] do
6:         d ← √M                                              ▷ max divisor
7:         while d ≥ 1 do
8:             for all j ∈ [1, d] do
9:                 if m[j] ≤ i then
10:                    N ← m[j] − (i − B + 1)                  ▷ offsetting
11:                    s[N] ← s[N] + j
12:                    if t[j] > √M then
13:                        s[N] ← s[N] + t[j]                  ▷ accumulate divisor's pair
14:                    end if
15:                    t[j] ← t[j] + 1
16:                    m[j] ← m[j] + j
17:                end if
18:            end for
19:            while m[d] > i do
20:                d ← d − 1
21:            end while
22:        end while
23:        s[j] ← 0, ∀j ∈ [0, B − 1]                           ▷ offload range before clearing
24:    end for
25: end procedure
```

-	SAS	Optimized Odd M-M
$X = 10^6$	0.08s	0.04s
$X = 10^7$	0.75s	0.36s
$X = 10^8$	8.20s	3.45s
$X = 10^9$	83.22s	47.17s

Table 4: Timings comparison between SAS algorithm and optimized odd M-M, sieving $s(\cdot)$ and $\sigma(\cdot)$ respectively to bound X.

SAS ($M = 10^8$)	Timing	$s[i]$ Buffer size
$B = 10^2$	18.14s	0.78 KiB
$B = 10^3$	9.23s	7.81 KiB
$B = 10^4$	8.43s	78.13 KiB
$B = 10^5$	8.62s	0.76 MiB
$B = 10^6$	8.70s	7.63 MiB
$B = 10^7$	17.38s	76.29 MiB
$B = 10^8$	18.91s	762.9 MiB

Table 5: Timings and s buffer sizes used by the SAS algorithm at different values of B.

5 Review of Related Works

5.1 Variant of a Theorem of Erdős on the Sum-of-Proper-Divisors Function

The authors of [**?**] are principally concerned with establishing upper and lower bounds on the density of numbers outside of the range of the *sum-of-unitary divisors* function, defined for all natural numbers by:

$$\sigma^*(n) = \sum_{\substack{d|n \\ \gcd(d,n/d)=1}} d.$$

This variant of the sum-of-divisors only sums divisors d that are co-prime with d/n. The authors also discuss previous analytic results on numbers outside of the range $\phi(n)$ and $s(n)$. $\phi(n)$ referring to *Euler's phi function* which counts positive integers $k \leq n$ such that k is co-prime with n, where k and n are co-prime if $\gcd(k, n) = 1$. Of particular interest to this project the authors also present a set of nearly linear algorithms to enumerate values outside the image of $s(n)$, $n - \phi(n)$, and $\sigma^*(n) - n$ within a bound.

The authors discuss a previously mentioned property that almost every odd number is in the image of $s(\cdot)$ thus when studying non-aliquots we need only consider even numbers. This follows from a stronger form of the Goldbach conjecture which states that all even $m \geq 8$ there exists distinct primes p and q such that $m = p + q$. Noting that the only proper divisors of a product of primes are those primes and 1,

$$s(pq) = p + q + 1 = m + 1$$

thus covering every odd number greater than or equal to 9. We have that 5 is the only odd non-aliquot as $s(4) = 3$, $s(8) = 7$, and $s(p) = 1$ where p is any prime.

The algorithm presented for the enumeration of non-aliquots within a bound is of key importance for this project, this will be referred to as the *Pomerance-Yang (P-Y) algorithm*. When enumerating non-aliquots it is useful to consider cases where $s(\cdot)$ maps to an even number.

Proposition 5.1. $s(n)$ is even if and only if

(i) $n = (2m+1)^2$, or

(ii) $2|n$ and $(n \neq m^2$ or $n \neq 2m^2)$.

We can observe that parity only flips when squares are encountered, it is worth outlining the intuition on why this is the case. Notice that a non-square necessarily must have a odd number of proper divisors as we include 1 and some number of factor pairs. However a square n must an even number of proper divisors as we include 1, $\sqrt{n}$, and some number of factor pairs. $(2m+1)^2$ will have an even count of strictly odd proper-divisors, the sum of such a set of numbers will have even parity. The even case seems to be a slightly more involved exercise, a discussion of this in the context of $\sigma(\cdot)$ is presented by [**?**].

The P-Y algorithm employs this property to split the computation of the even image of $s(\cdot)$ into distinct cases. For the even case the authors construct a recurrence from the following proposition.

Proposition 5.2. Let j be a positive integer and m be an odd positive integer, the following statements hold:

(i) $s(2m) = 3\sigma(m) - 2m$,

(ii) $s(2^{j+1}m) = 2s(2^j m) + \sigma(m)$.

The multiplicative property of the $\sigma(\cdot)$ [**?**] gives context to how these statements where constructed. For the odd case the image of numbers p^2 for some prime $p < x$ are detected within the same loop as the recurrence while m^2 for some odd composite $m < x^{2/3}$ are computed separately. Non-aliquots are numbers within the range that have not been touched by the enumeration.

Since the P-Y algorithm for $s(\cdot)$ was described in comparison to similar algorithms I will outline this version as a stand-alone algorithm.

Algorithm 3 P-Y Algorithm to Enumerate Non-Aliquots

```
1: procedure ENUMERATE NON ALIQUOTS(x)
2:     f[i] ← 1, ∀i ∈ [1, x]
3:     for all odd m ∈ [1, x] do
4:         if 2|σ(m) then
5:             t ← 3σ(m) − 2m
6:             while t ≤ x do
7:                 f[t] ← 0
8:                 t ← 2t + σ(m)
9:             end while
10:        end if
11:        if σ(m) = m + 1 then                    ▷ if m is prime
12:            f[m + 1] ← 0
13:        end if
14:    end for
15:    for all odd composite m ∈ [1, x^{2/3}) do
16:        if s(m^2) ≤ x then
17:            f[s(m^2)] ← 0
18:        end if
19:    end for
20:    return f
21: end procedure
```

5.2 Some Problems of Erdős on the Sum-of-Divisors Function

[**?**] tackle 3 problems regarding the sum-of-divisors function: bounding the number of aliquot reversals, bounding the number friendly pairs, and of specific interest developing a heuristic model for the density of non-aliquots. Contextually this paper largely engages with problems proposed by Erdős with an eye for relating the results back to the Guy-Selfridge / Catalan-Dickson conjectures.

Aliquot sequences have a tendency to preserve abundancy or deficiency, an aliquot reversal a number where this tendency does not hold. These reversals are of key importance when considering amicable pairs or sociable numbers more generally, these cycles require a reversal of abundancy to form. Take the following amicable pair for example:

$$s(220) = 284 \ \textit{up-down reversal}$$
$$s(284) = 220 \ \textit{down-up reversal}$$

The authors note a result of Erdős who by proving that there are few up-down reversals was able to establish that the numbers involved in amicable pairs (later extended to sociable numbers) have a vanishing asymptotic density. With this an analog of the Guy-Selfridge heuristic argument for sequence unboundedness is presented:

- **if** aliquot sequences tend to preserve parity, and
- **if** sociable cycles are of vanishing density, and
- **since** a sequence is only bounded if it enters a cycle or terminates on a prime;
- **then** even sequences tend to be unbounded.

However the authors point out the result of [**?**] who demonstrated that on average even numbers tend to be slightly deficient, as such under this argument we would expect even sequences to be bounded.

While not a direct response to the text this line-of-reasoning did prompt me to consider the purpose of this style of heuristic argument more generally. If we assume that $\mathcal{S}(2n) < 2n$ we would not have a proof that all even sequences will terminate. For demonstrative purposes consider a toy function defined by:

$$r(n) = \begin{cases} 10n & \text{if } n = 10^m \\ 1 & \text{otherwise} \end{cases}$$

Sequences generated by iterating this function will be unbounded only if n is a power of 10. Based on some quick numerical estimation it seems the geometric mean of $r(n)/n$ tends towards 0. This demonstrates the possibility that some subset of sequences can be unbounded while on average sequences tend to decrease. I am not sure if it is possible to construct an analogous example where the geometric mean of $r(n)/n$ is greater than 1 but every sequence is bounded.

30 and 120 are examples of numbers that share the same ratio $\sigma(n)/n$, such numbers are known as *friendly pairs*,

$$\frac{\sigma(30)}{30} = \frac{12}{5} = \frac{\sigma(120)}{120}.$$

The authors prove bounds on how many numbers satisfy a slightly stronger condition of *primitive friendly pairs*. While not a direct response to the text it is interesting to note that the concept of friendly pairs and more generally *friendly clubs* might be useful when storing a computed range of $\sigma(n)$. Storing $\sigma(n)/n$ instead could possibly reduce the number of bits needed and the presence of friendly clubs introduces repetition which would improve data compression ratios.

Most relevant for my project the authors construct a heuristic model for the density of non-aliquots,

$$\Delta = \lim_{y \to \infty} \frac{1}{\log y} \sum_{\substack{a \leq y \\ 2 \mid a}} \frac{1}{a} \mathrm{e}^{-a/s(a)}.$$

The bulk of my previous work term was spent in generalizing this result into Conjecture 1.2. In brief this model was constructed by setting up the relevant

criteria and assumptions to apply the "balls and bins" statistics problem to estimate the number of non-aliquots. We consider the results of $s(n)$ as the thrown balls and some well defined subset of the even numbers as the bins. Say you have n bins and toss m balls, we can compute the probable number of empty bins after this process with:

$$\#[\text{empty bins}] = n(1 - \frac{1}{n})^m.$$

The number of "empty bins" thus estimates the number of non-aliquots.

5.3 Numerical and Statistical Analysis of Aliquot Sequences

[1] In [**?**] the weighted geometric mean of $s(n)/n$ is computed with each ratio weighted by $\#s^{-1}(n)$, the idea being that numbers with many parents have a larger impact on the average behaviour of sequences. This method gives 0 weight to $s(n)/n$ for $\#s^{-1}(n) = 0$, however non-aliquot numbers can be thought of as the start of sequences rather than numbers excluded from sequences. This is necessarily true, if some number n has $\#s^{-1}(n) > 0$ then there must be some preceding number in the same sequence, an non-aliquot is where this recursion terminates. Consider the sequence originating at the non-aliquot 52:

$$s(52) \to s(46) \to s(26) \to s(16) \to s(15) \to s(9) \to s(4) \to s(3) \to s(1)$$

It seems to me that $s(52)/52$ has no less impact on the termination than any other number in the sequence. Perhaps non-aliquots should be considered with weight 1 when computing such an average.

6 Conclusion

My project on computationally and heuristically quantifying the densities of k-parent aliquot numbers fits best into existing literature when discussed in the context of the Guy-Selfridge conjecture. In this approach the conjecture is best viewed as a way to inspire new and creative mathematics rather than a statement that urgently requires a proof. With this mindset I hope this work might be able to add some interest into the heuristic discussion of Guy-Selfridge.

Elaborating the discussion around k-parent numbers could enable further explorations into more general *aliquot genealogies*, just as the generalization of non-aliquots enabled this project. By this I am referring to the idea an aliquot number also has some number of *grand-parents* or in other words the number of sequences tributary to a number 2 pre-images removed, Figure 1 provides an illustration of this concept. It is possible that such considerations could provide an interesting way to model sequence behaviour.

[1] My initial draft at reviewing [**?**] was largely edited into the motivation section of this paper. Unfortunately due to time constraints I was unable finish this review, however I believe this isolated point is interesting enough to leave in.

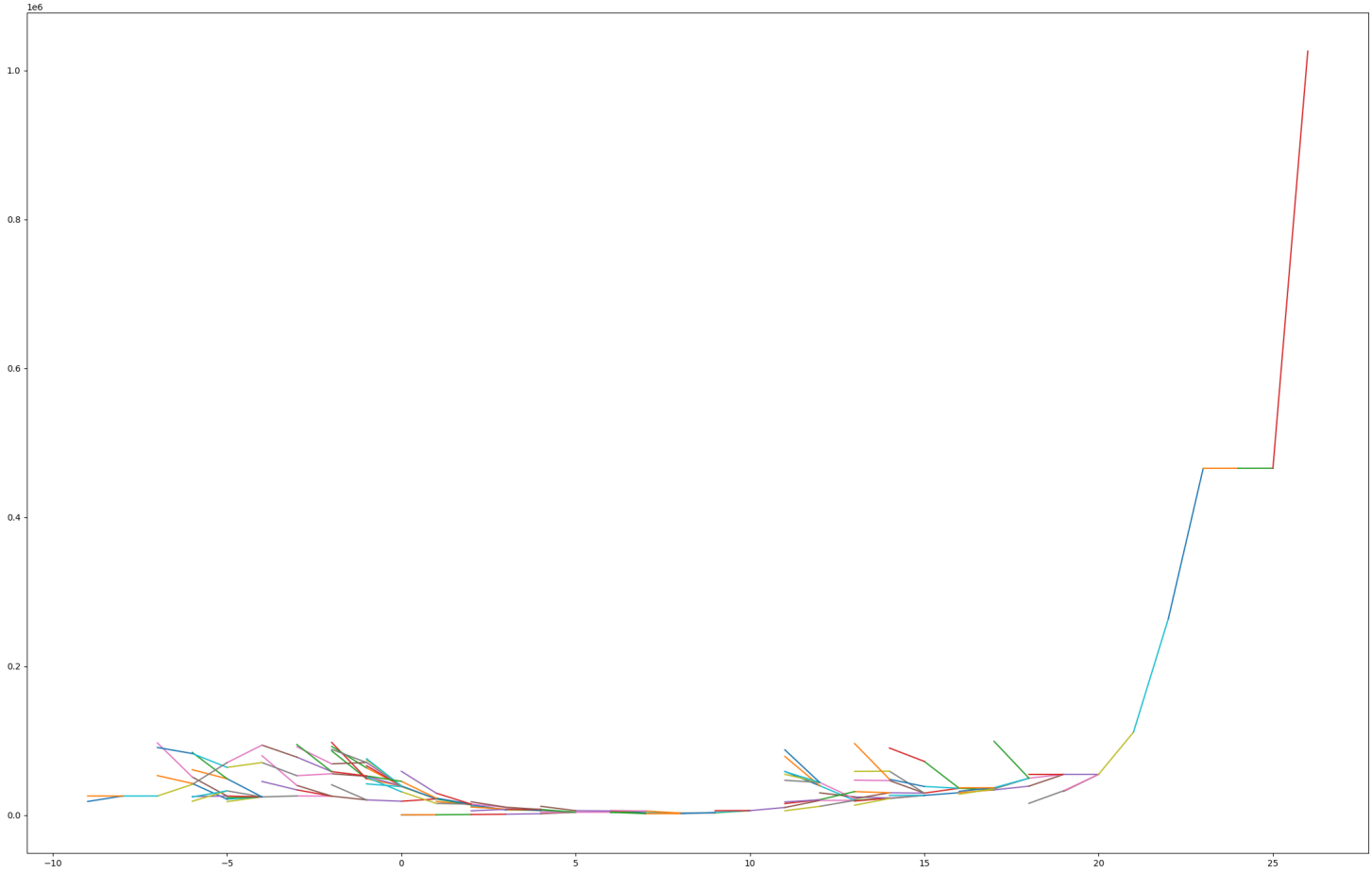

Figure 1: Recursive plot of the 276 sequence's aliquot genealogy, the x axis is the number of iterations before or after 276, the y axis is the value of the aliquot number.