Rose-Hulman Undergraduate Mathematics Journal

Volume 20
Issue 1

Article 4

New Experimental Investigations for the 3x+1 Problem: The Binary Projection of the Collatz Map

Benjamin Bairrington
i_is_confused@yahoo.com

Aaron Okano

Rose-Hulman Undergraduate Mathematics Journal
VOLUME 20, ISSUE 1, 2019

New Experimental Investigations Concerning the $3x+1$ Problem: The Binary Projection of the Collatz Map

By *Benjamin Bairrington* and *Aaron Okano*

Abstract. The $3x+1$ Problem, or the Collatz Conjecture, was initially developed in the early 1930's. It has remained unsolved for over eighty years. Throughout its history, traditional methods of mathematical problem solving have only succeeded in proving heuristic properties of the mapping. Because the problem has proven to be so difficult to solve, many think it might be undecidable. In this paper, we briefly follow the history of the $3x+1$ Problem from its creation in the 1930's to the modern day. Its history is tied to the development of the Cosper Algorithm, which maps binary sequences into integer families. Then we discuss the pseudo-code which the Cosper Algorithm is based upon. A simple example is provided to demonstrate the Cosper Algorithm. Afterwards, the generalized $3x+k$ Problem is considered yielding two definitions: k-dependent and k-independent cycles. Finally, some images are provided of various k-dependent cycles.

1 Introduction

1.1 Roadmap

Within the mathematical branch of Number Theory lies the study of sequences generated by arithmetic functions. Usually, an arithmetic function is defined by taking a natural number (or in some cases an integer) as an input, and depending on some specified cases, yields different outputs. The process is repeated, replacing the original input with the output thus creating a sequence of numbers. Natural questions arise such as whether a sequence halts (becomes undefined), crosses through some value, or is bounded in some way. The $3x+1$ Problem (see Definition 1) is one such number-theoretic problem.

The $3x+1$ function maps natural numbers to natural numbers with only two cases. If an input is odd multiply it by three and add one. If an input is even divide it by two. What piques the interest of many toward this problem is the fact that it has been extraordinarily

Mathematics Subject Classification. Primary 11B75; Secondary: 11Y16, 11Y70
Keywords. Collatz, $3x+1$, inverse map, number theory algorithms

difficult to answer any of the natural questions. In fact, there is a degree of suspicion that answering such questions may be impossible. This has not stopped a following of researchers, scientists, recreational mathematicians, and students from examining it from all angles. Up to this time, very little progress has been made using conventional mathematical tools.

This paper is intended to present a new mathematical tool to study the $3x+1$ Problem. The premise begins by considering a new function X (see Definition 3). This function X takes the input of the $3x+1$ function and assigns it either a one or a zero depending on its parity. In this way a binary sequence can be created from the sequential outputs of the $3x+1$ Problem. Such binary sequences can be examined under the same natural questions. Moreover, there are questions (and answers) that can be stipulated regarding binary sequences that would be difficult to pose to the $3x+1$ Problem itself. However, the full scope of study regarding binary sequences would be moot if in fact there wasn't a method to map binary sequences back into the $3x+1$ Problem.

The Cosper Algorithm, originally developed by the authors, is a method which binary sequences can be mapped back into the $3x+1$ Problem. It serves as a computational X^{-1} function. The Cosper Algorithm may serve a variety of purposes. A reader's interests could lie in determining when certain number patterns appear, how the family of integers shrinks when a binary sequence is extended, where rational cycles may exist, what kinds of integers satisfy certain parity conditions, among other related questions. The Cosper Algorithm is a means by which these related questions could be answered.

Anyone looking for new methods and techniques to approach the $3x+1$ Problem may have interest in this paper, and the Cosper Algorithm. This paper presents the construction of the algorithm, and provides a demo of its usage. Afterwards, some observations are made regarding cycle structures which are more accessible to interpret in the form of binary sequences as opposed to sequences of integers. The paper is presented in the following way: Section 1.1 is the abstract of the paper. Section 1.2 is the content summary and roadmap of the paper. Section 1.3 is a historical summary of the $3x+1$ Problem. Section 1.4 is a summary of modern progress regarding the $3x+1$ Problem. Section 2.1 is a description of the purpose of the Cosper Algorithm. Section 2.2 details the program function by function. Section 3.1 is some observation of the $3x+k$ map. Section 3.2 is visuals of various periodic sequences under chosen $3x+k$ mappings. Section 4 is the conclusion of the paper's primary content. Section 5 is acknowledgements of people who were generous in their time, knowledge, and spirit. Section 6.1 is discussion of the scope for which presentation of the Cosper Algorithm is appropriate. Section 6.2 is an operational summary of the Cosper Algorithm. Section 6.3 is some computational limits of the Cosper Algorithm. Section 6.4 is a standard operation procedure of the Cosper Algorithm implemented in Python. Section 6.5 is an example operation. Section 6.6 is known errors of the Cosper Algorithm. Section 6.7 is the closing remarks. The paper closes with its bibliography.

1.2 History

When the $3x+1$ Problem first came into print in 1971, it had already become quite the intellectual attraction. Having traveled by word of mouth, it had spread from its home in the University of Berlin to Los Alamos National Laboratory in the United States. It had been examined by some of the best mathematicians of the day, yielding results that fell quite short of any meaningful resolution. Dr. Coxeter writes in his 1971 lecture notes on cyclic sequences that, “sometimes the best way to tell a tale is back-wards." [4] This suits the pursuit of the $3x+1$ Problem as well. From its infancy as the brainchild of Dr. Collatz to its international acclaim, progress on the $3x+1$ Problem has been marked not by breaking it down, but by building it up.

The $3x+1$ Problem has many names, although perhaps one of the most familiar names is the Collatz Conjecture. Dr. Collatz was born July 6, 1910, in Arnsberg Westphalia, Germany [18]. His interests as a student were in mathematics and physics, leading him to enter the University of Greifswald in 1928 [13]. As it was customary to do at the time, Dr. Collatz continued his undergraduate education by attending different Universities [11]. In 1929, he had moved to Göttingen University. By 1930 he had passed through Munich University, and was settled in the University of Berlin [11]. It was during his stay in Berlin that he took his first course in Graph Theory [11]. Dr. Collatz had acquired a growing interest in number-theoretic functions throughout his intellectual tour, and his exposure to Graph Theory led him to attempt a novel experiment.

In his own words, he considered first the simple function $f(n) = n+1$. He assigned the initial input as the root of a tree, edges the respective compositions of f, and nodes the functional output on each composition [2]. The divergent trajectory of f can easily be seen as a path starting from the root and moving arbitrarily away for each consecutive son of the tree (see Figure 1). It is not hard to determine from the graph that there are no cycles and that f diverges for all initial inputs.

Figure 1: Path generated by the number-theoretic function $f(n) = n+1$.

A similar example is the number-theoretic function of form $f(n) = n - g(n)$, for $g(n)$ being the greatest proper divisor of n [2]. In this example, if we consider multiple starting values n, all greater than one, each root can be reinterpreted as a leaf for a tree rooted at one (see Figure 2).

Taking an example one step further, consider the number-theoretic function of the form,

$$f(n) := \begin{cases} 3n, n \text{ prime} \\ \text{sum of all proper divisors greater than one, else} \end{cases} . \tag{1}$$

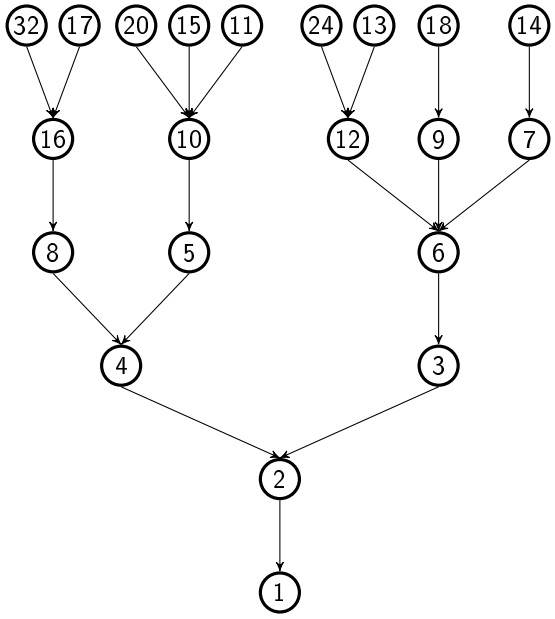

Figure 2: Tree generated by the number-theoretic function $f(n) = n - g(n)$.

We find the graphical structure is similar to the previous example (see Figure 3), with the notable distinction that it is no longer a tree. To begin, the overall graph is composed of continuous components, each disjoint from each other. Moreover, following the trajectory (arrows) from each n, we find ourselves caught in a loop. These loops are called cycles and are unique from the previous two examples. Cycle structures of number-theoretic functions attracted Dr. Collatz's curiosity [2]. Wanting to know more about cycles, he proceeded to explore number-theoretic functions inductively in a direction which may seem quite familiar.

Dr. Collatz wanted to create cycles like the above example, but using only elementary calculations. In essence he observed that a cycle would only be closed if the number-theoretic function $f(n) > n$ for some n, and $f(n) < n$ for other n [2]. His first thought was to define $f(n) = n/2$ if n was even, and $f(n) = n+1$ if n is odd. Indeed, it does fit the theoretical picture for a cycle, but only trivially for the cycle (1,2). The next logical step was to increase the size of n. Defining $f(n) = n/2$ if n was even, and $f(n) = 2n+1$ if n is odd again meets the theoretical framework. Unfortunately though, by the defined parity, all initial odd integers diverge creating no cycle. In one more attempt Dr. Collatz defined $f(n) = n/2$ if n was even, and $f(n) = 3n+1$ if n is odd (see Equation 2, and Figure 4). In a purely innocent inductive process, Dr. Collatz had stumbled upon the $3x+1$ Problem.

$$f(n) := \begin{cases} 3n+1, n \equiv 1 \mod 2, \\ \frac{n}{2}, n \equiv 0 \mod 2 \end{cases} . \tag{2}$$

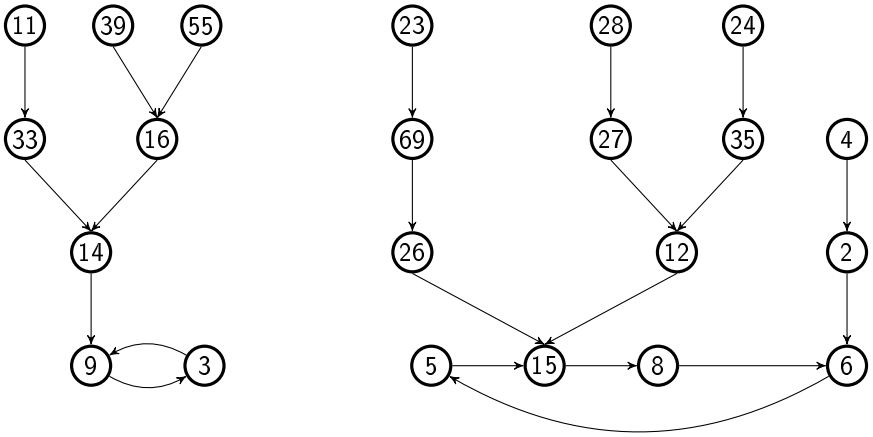

Figure 3: Graph generated by the number-theoretic function defined in Equation 1.

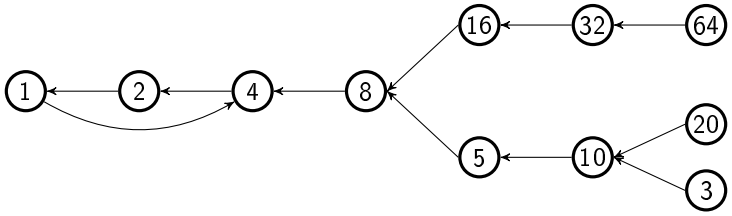

Figure 4: Graph generated by the Collatz Conjecture.

Readily one cycle appears with the trajectory 4-2-1-... However it was not clear whether this was the only cycle, or perhaps some seed integers would diverge. However, his contribution largely ends there. To say his interest in the $3x+1$ Problem had waned in the years afterward would be misleading. Dr. Collatz had a strict moral code regarding mathematical research. He believed mathematicians had a responsibility to apply their results to real world phenomena [13]. This is reflected in his 1934 dissertation entitled, "Difference Methods with Higher Approximation for Linear Differential Equations" [18]. He remained in the world of applied math, and in 1935 was appointed to an assistantship in the Institute for Technical Mechanics in Karlsruhe [18]. This led to his Habilitation Thesis in 1937 on "Convergence Proofs and Error Analysis for Difference Methods for Eigenvalue Problems Associated with Differential Equations of Second and Fourth Orders" [18]. His academic career proceeded by being appointed Privat-Dozent at the Karlsruhe Institute of Technology in 1938 [18]. He left in 1943 to hold a chair at the Technical University of Hanover [13]. It was in this position that he arrived in Cambridge, Massachusetts in 1950 for the International Congress of Mathematics [10].

At the International Congress of Mathematics, Dr. Collatz told many of his colleagues about the $3x+1$ Problem. Most notably were Dr. Coxeter, Dr. Kakutani, and Dr. Ulam [11]. In 1952 Dr. Brian Thwaites independently created the $3x+1$ Problem [10]. Additionally

in 1952, Dr. Collatz accepted a position at Hamburg University and presented the $3x+1$ Problem to his colleague Dr. Hasse [18]. It is through these four (Dr. Coxeter, Dr. Kakutani, Dr. Ulam, and Dr. Hasse) that the problem would spread by word of mouth to many academic institutions worldwide. This process began when Dr. Hasse presented the $3x+1$ Problem to Syracuse University, leading to the nick-names *The Syracuse Problem* as well as *Hasse's Algorithm* [11]. In 1960 Dr. Kakutani presented the problem at Yale and the University of Chicago leading to the nick-name *Kakutani's Problem* [10]. From there it went to the Massachusetts Institute of Technology where the $3x+1$ Problem was checked for all starting values less than sixty million [7].

It was not until 1971 when the $3x+1$ Problem first appeared in print in a set of memorial lecture notes by Dr. H.S.M. Coxeter [4]. Nevertheless, it remained an underground problem appearing as a novelty in Scientific American in 1972 [7]. In 1975 Dr. Hasse wrote about the problem [9], again putting it into print. In the mid-seventies Dr. Ulam spread the problem to Los Alamos National Laboratory [11] where it was received by Dr. Everett who wrote about the problem in 1977 [6]. Around this time the $3x+1$ Problem become an internationally known problem. The $3x+1$ Problem remained unsolved upon Dr. Collatz's death on September 26, 1990 [13].

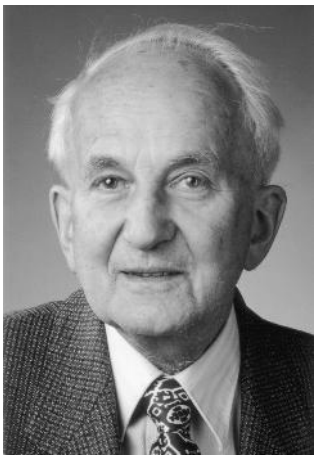

Figure 5: Dr. Collatz, compliments of University of Hamburg.

In the modern day, there are much more powerful methods to test the truthfulness of the $3x+1$ Problem. This includes more advanced computers as well as mathematical methods. In 2010 Dr. Tomás Oliveira e Silva proved the conjecture up to starting values $n \leq 20 \times 2^{58}$ [14]. Finally in 2011, Dr. Gerhald Opher, a PhD student of Dr. Collatz, attempted a solution for the $3x+1$ Problem using holomorphic mapping that yielded a partial proof [15]. The $3x+1$ Problem has gone a long way from its beginning in Dr. Collatz's undergraduate notebooks, and its lack of proof demonstrates only that there is much more mathematics to be created to truly understand the problem.

1.3 Modern Progress

Although the terms $3x+1$ and Collatz are used interchangeably to discuss the problem, there is a subtle difference regarding their respective integer mappings. The former is denoted with a T and the latter with a C. Both are expressed below in Equation 3 of the following definition:

Definition 1.1. The following mappings C (Collatz) and T ($3x+1$) map the integers to

the integers in the following form:

$$\mathrm{C}(n) := \begin{cases} 3n+1, n \equiv 1 \mod 2, \\ \frac{n}{2}, n \equiv 0 \mod 2 \end{cases}, \mathrm{T}(n) := \begin{cases} \frac{3n+1}{2}, n \equiv 1 \mod 2, \\ \frac{n}{2}, n \equiv 0 \mod 2 \end{cases}. \tag{3}$$

The conjecture regarding either T or C is whether for every positive integer n, there exists a positive integer exponent k such that $\mathrm{C}^k(n) = 1$, or $\mathrm{T}^k(n) = 1$, respectively. A simple example of this problem is to consider the starting (seed) number $n = 3$. We find the path generated by each of these integers proceeds as follows (see Figure 6).

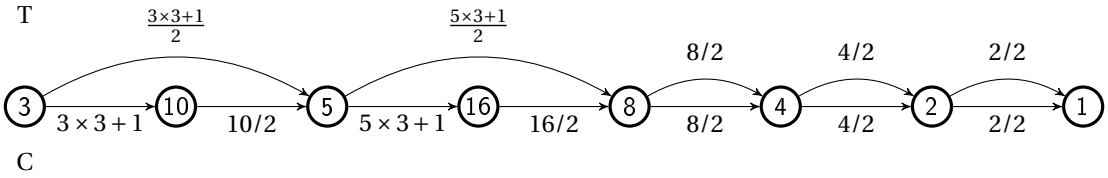

Figure 6: Paths created by the $3x+1$ (T) and Collatz (C) integer mappings.

When considering the modern progress a natural beginning is to ask how difficult the problem is. One could start by measuring the complexity of its problem statement. The Collatz Conjecture hypothesizes that for every natural number n there exists an exponent k such that $\mathrm{C}^k(n) = 1$. This makes the conjecture a Π_2 statement that can be written in the form: $(\forall n)(\exists k)(\mathrm{C}^k(n) = 1)$ [12].

There are other variants of 'Collatz-like' problems that are sufficiently complex to be considered undecidable. One such example is the following mapping from the integers to integers (Equation 4),

$$\begin{aligned} &g : \mathbb{Z} \to \mathbb{Z} \\ &g(n) = a_i n + b_i, \ n \equiv i \pmod{p}, \end{aligned} \tag{4}$$

with $a_i, b_i \in \mathbb{Q}$, and $g(n)$ always yielding an integer. The conjecture regarding Equation 4 was similar to Collatz. The conjecture is that for every map g and every integer n there exists an exponent k such that $g^k(n) = 1$ [16]. This was demonstrated to be undecidable by comparing it to fractional-linear mapping similar to Equation 4 with all $b_i = 0$ [12]. The fractional-linear mapping itself is undecidable because it is a special case of the FRACTRAN algorithm, which had been demonstrated to be a universal computer. Hence, if there were a means by which a k could be found given a map g and integer n, then FRACTRAN could resolve the Halting Problem, which would be a contradiction [12]. A similar 'Collatz-like' mapping appears in Equation 5:

$$\begin{aligned} &f : \mathbb{Z} \to \mathbb{Z} \\ &f(n) = \begin{cases} \frac{n}{2}, \ n \equiv 0 \pmod 2 \\ 3n + t, \ n \in \mathrm{A}_t \end{cases}, \end{aligned} \tag{5}$$

where A_t is a periodic set, and t = -9,-8,...,8,9 [6]. Even in this case, a modularity argument demonstrates that it is complex enough to resolve the Halting Problem and is undecidable [12].

Unfortunately, this does not readily fit to disprove the $3x+1$ conjecture. Modern evidence points towards the possibility that there may as of yet be a conclusive answer. To begin, consider the $3x+1$ map T defined in Equation 3 possessing many interesting properties. For example, consider the characteristic mapping χ of T defined in Equation 6 in the following definition.

Definition 1.2. The characteristic mapping χ takes N iterative steps (indexed by the variable i) of the T mapping to a boolean variable determined by the parity of the value of the function at each step.

$$\chi_i(n) = \begin{cases} 1, \ \mathrm{T}^i(n) \equiv 1 \pmod 2 \\ 0, \ \mathrm{T}^i(n) \equiv 0 \pmod 2 \end{cases}. \tag{6}$$
$$n \to \{\chi_0, \chi_1, \chi_2, ..., \chi_{\mathrm{N}-1}\}$$

For example let N = 3:

n	$\mathrm{T}^i(n)$	$\{\chi_i\}$
4	$[4 \to 2 \to 1]$	001
2	$[2 \to 1 \to 2]$	010
6	$[6 \to 3 \to 5]$	011
5	$[5 \to 8 \to 4]$	100
1	$[1 \to 2 \to 1]$	101
3	$[3 \to 5 \to 8]$	110
7	$[7 \to 11 \to 17]$	111

Figure 7: Finite Binary expansion of integers less than 8

Induction can prove that the binary sequence of length N is unique for integers $n < 2^{\mathrm{N}}$ [6]. Hence, there exists an injective mapping between the integers $n < 2^{\mathrm{N}}$ and binary sequences of length N [6]. Furthermore from a statistical standpoint, for every $n < 2^{\mathrm{N}}$, the probability that each inductive step of χ_i being a 1 or 0 approaches one-half (like a coin flip) for large N [5]. This property allows for the exploration of an 'average growth rate' of the T mapping. Assuming that probability is exactly one-half [16], then there is a fifty percent chance the integer increases by a factor of 3/2 or decrease by 1/2. From each inductive step, the same probability holds. The average growth rate of a trajectory N − 1 odd numbers appears as a product of the form: $(3/2)^{\mathrm{N}/2} \times (3/4)^{\mathrm{N}/2} \times (3/8)^{\mathrm{N}/2} \times ...$ converging to 3/4 as N goes to infinity [16].

It is much easier to generate a computer program that follows algebraic instructions than logical arguments. Hence, many have restated the above definitions so they can

be explored with a computer. One could consider addressing classes of integers which converge under different kinds of patterns [3]. Various trajectories can be reversed by generating a power series whose coefficients are determined by Equation 6. These convergence classes create upper and lower bounds on trajectories which determine whether an integer may diverge, converge, or behave periodically [1]. Similarly, many trajectories can be bound within an $n \times n$ matrix. Iterations of the Collatz mapping are represented by matrix products [1]. The problem regarding periodic cycles can be reduced to subsets of the integers [3].

2 The Cosper Algorithm

2.1 Premise

The Cosper Algorithm is a useful tool allowing the Collatz Conjecture to be studied from a vantage inaccessible from either integers rooted to one, or numbers tending to be arbitrary large. For example, one may like to study the trajectory of 27. Starting from 1 and moving 'backward' along the graph of the Collatz Conjecture (see Figure 4) would have to take 111 steps (assuming no mistakes were made). As discussed above, observing the trajectory of arbitrarily large numbers has yielded heuristic properties. The Cosper Algorithm was written by the authors of this paper with the purpose of producing an inverse mapping (X^{-1}) which maps binary sequences to the integers following the definition of the mapping C (see Definition 3). In this section, we present the Cosper Algorithm which takes finite binary sequences as an input, and outputs a family of odd positive integers. This family of integers represents all integers who, under the characteristic mapping X, yield a binary sequence matching the Cosper input on every iterative step. The family is the maximal set of integers which match the binary sequence. This algorithm is equipped with a simple graphical user interface and is implemented using Python. We can consider a road map of the algorithm below in Figure 8.

Definition 2.1. The characteristic mapping X takes N iterative steps (indexed by the variable i) of the C mapping to a boolean variable determined by the parity of the value of the function at each step.

$$\begin{gathered} X_i(n) = \begin{cases} 1, \ C^i(n) \equiv 1 \pmod 2 \\ 0, \ C^i(n) \equiv 0 \pmod 2 \end{cases}. \\ n \to \{X_0, X_1, X_2, ..., X_{N-1}\} \end{gathered} \tag{7}$$

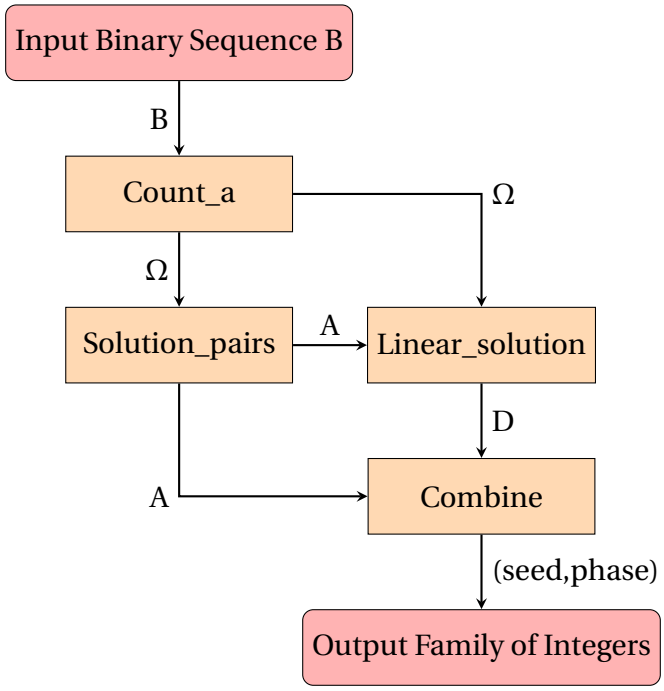

Figure 8: Flowchart of the Cosper Algorithm

2.2 Pseudo Code

The `count_a` subprogram accepts a finite binary sequence B starting with the following requirements:

1. The binary sequence B starts with a 1.
2. Every 1 within the binary sequence B is immediately followed by no less than a single 0.
3. The binary sequence B ends with at least a single 0.

It proceeds to read each element b_i of B from left to right. It outputs an array Ω of pairs (k, j) for each consecutive pair $(10)^k + (0)^j$. Each pair (k, j) is generated in the same way. For each consecutive instance $(b_i, b_{i+1}) = (1,0)$, k is incremented by one, and i by two. In the case $b_i = 0$, k is fixed while j and i are incremented by one. If $b_i \neq 0$ then again $(b_i, b_{i+1}) = (1,0)$. Before $(1,0)$ is counted (but after it is read), the pair (k, j) is stored into Ω and the values of k and j are reset. The next (k, j) begins by counting this new $(1,0)$. This process is repeated until the entire binary sequence is read.

Example: Consider the binary sequence B = 10100001000. From left to right we see two (1,0), followed by three (0), another (1,0), and finally two (0). Together they generate $\Omega = ((2,3),(1,2))$, which is the output of `count_a`.

The `solution_pairs` subprogram accepts as an input an array whose elements are pairs of positive integers $((k_1, j_1), ..., (k_n, j_n))$. In the Cosper Program, the subprogram `solution_pairs` occurs sequentially after `count_a`, accepting the output Ω as its input. `Solution_pairs` outputs an array A whose elements are pairs of numbers $((seed_1, phase_1), ..., (seed_n, phase_n))$. Each element in A corresponds directly to an element of Ω in the following manner:

$$\texttt{count\_a}\big(\{\mathrm{X}^{k+j}(seed + m \times phase)\}\big) = (k, j), \text{for } m \in \mathbb{Z}_+. \tag{8}$$

That is, each pair $(seed, phase)$ is the integer family representing the binary subsequence of B whose (10), (0) count corresponds to (k, j) in Ω (whereas by hypothesis B obeys the stated provisions).

The pairs (k, j) in Ω are read in `solution_pairs` sequentially. From Equation 7, we are looking for a pair of integers $(seed, phase)$ which satisfies the equality. This can be reduced down to a problem of a single variable n in the following form:

$$2^j | 3^k - 1 + m \times 2 \times 3^k, \text{for some } m \in \mathbb{Z}_+. \tag{9}$$

Effectively, this subprogram determines the mapping of binary sequences for the simplest nontrivial structure. Resolving each pairwise input (k, j) determines the mapping piece-by-piece. This result is true by considering the following propositions (1 and 2):

Proposition 2.2. *The Characteristic Trajectory of a natural number n of the form $n = 2^k - 1 + 2^{k+1}m$, for $k, m \in \mathbb{Z}_+$ begins with a binary sequence 1010... of k-repetitions of 1,0.*

Proof. Consider a natural number of the form $n = 2^k - 1 + 2^{k+1}m$. This number is odd, so $\mathrm{X}_0(n) = 1$. Let us count this as the first step. Applying C to n yields the following expression:

$$\begin{aligned} \mathrm{C}(n) = \mathrm{C}(2^k - 1 + 2^{k+1}m) &= 3 \times 2^k - 3 + 3 \times 2^{k+1} + 1 \\ &= 2(3 \times 2^{k-1} - 1 + 3 \times 2^k m), \end{aligned}$$

which is even implying $\mathrm{X}_1(n) = 0$. Applying C one more time yields the value:

$$\mathrm{C}^2(n) = \mathrm{C}(2(3 \times 2^{k-1} - 1 + 3 \times 2^k m)) = 3^1 \times 2^{k-1} - 1 + 3^1 \times 2^k m.$$

This expression is again odd, implying that $\mathrm{X}_2(n) = 1$. Observe also, that the above expression has replaced a factor of 2 with a factor of 3 in two steps of C. The expression is odd for the same reason as the original n was odd, with the distinction that the difference is taken between two multiples of a power of two (now being of degree $k-1$) and one.

Taking the third, and fourth steps of k we observe that:

$$C^3(n) = 2(3^2 \times 2^{k-2} - 1 + 3^2 \times 2^{k-1} m),\ X_3(n) = 0$$
$$C^4(n) = 3^2 \times 2^{k-2} - 1 + 3^2 \times 2^{k-1} m,\ X_4(n) = 1.$$

The pattern is clear that every second step yields an odd natural number which swaps a power of two with a power of three. Continuing to steps $2k-1$ and $2k$ we find that:

$$C^{2k-1}(n) = 2(3^k \times 2^{k-k} - 1 + 3^k \times 2^{k+1-k} m) = 2(3^k - 1 + 2 \times 3^k m),\ X_{2k-1}(n) = 0$$
$$C^{2k}(n) = 3^k - 1 + 2 \times 3^k m.$$

The value of $X_{2k}(n)$ is zero regardless of the value m, as the above equation is even for every m. Hence at the end of k repetitions of $(1,0)$ the pattern stops.

□

Proposition 2.3. *For a given natural number n, its binary sequence* B_k *(constructed by* X*) listed up to step k is identical to the binary sequence* B_k^* *for a natural number* $n + 2^j m$*, for j is the number of zeros in the binary sequence up to step k and* $m \in \mathbb{Z}_+$.

Proof. Consider first the case which $m = 0$, of which corresponds to the integer n. By the hypothesis, n has binary sequence B. For all steps k^* less than k, the parity of 2^j will remain even. As the parity of an even number does not affect the parity of an odd number, then the parity of n is independent of 2^j. Hence, $X_{k^*}(n) = X_{k^*}(n + 2^j)$ up to step k. □

By Propositions 1 and 2, we know that Equation 8 matches k in (k, j) for any n. To satisfy j we must also satisfy Equation 9. To reduce that equation even further, consider the following rearrangement using a dummy variable p.

$$\begin{aligned} 2^j p - 2 \times 3^k n &= 3^k - 1 \\ 2^j p &= 3^k(2n+1) - 1 \\ 0 &\equiv 3^k(2n+1) - 1 \mod (2^j) \\ \implies 1 &\equiv 3^k(2n+1) \mod (2^j). \end{aligned} \tag{10}$$

To proceed from here, observe that 3^k and 2^j are co-prime concerning each other. This allows us to apply Euler's Theorem to reduce the congruence even further. As a brief review, consider the following definition of the ϕ-function (Definitions 3 and 4) followed by Euler's Theorem (Theorem 1) [17].

Definition 2.4. Euler's Phi Function ϕ (or the totient function) [17], is the number-theoretic function which for a positive integer n outputs the number of positive integers less than and co-prime to n.

Definition 2.5. The reduced system of residues modulo n [17], is the set of mutually incongruent integers of size $\phi(n)$ of whom are all co-prime concerning n. In the case n is prime, the size of the set is $n-1$.

Theorem 2.6. *(Euler's Theorem) [17] If a and n are two relatively prime integers, then*

$$a^{\phi(n)} \equiv 1 \pmod{n}.$$

Notably in our case, as 2 is prime, then $\phi(2^j) = 2^j - 2^{j-1} = 2^{j-1}$. Therefore by Euler's Theorem:

$$\left(3^k\right)^{\phi(2^j)} = 3^{k\phi(2^j)} = 3^{k2^{j-1}} \equiv 1 \pmod{2^j}. \tag{11}$$

Replacing the left-hand side of the congruence relation in Equation 10 with the left hand side of Equation 11 yields:

$$2n+1 \equiv 3^{k(2^{j-1}-1)} \pmod{2^j}. \tag{12}$$

Thus, the original division problem has been reduced down to a problem of a single variable. From here, the coefficient n can be solved for by modular exponentiation of powers of three. Starting from one, multiplying by three, and taking the modulus 2^j for $k(2^{j-1}-1)$ steps yields $2n+1$. The value of n is solved by subtracting one and dividing by two. This process is repeated for all pairs (k, j) in Ω yielding pairs (seed,phase) stored in array A which correspond to the numbers that satisfy Equations 7 and 8 of the form:

$$(2^k - 1 + n2^{k+1}, 2^{k+j}). \tag{13}$$

This completes `solution_pairs`.

Example: Consider the binary sequence 1010000. $\Omega = ((2,3))$. Here, $k = 2$ and $j = 3$. Therefore $2^3 = 8$, and $k(2^{j-1}-1) = 2 \times (2^{3-1}-1) = 6$. Modular exponentiation for six steps will find that $3^6 \equiv 1 \mod (8)$. Therefore $2n+1 = 1$ implying $n = 0$. This yields the pair ((3,32)) for A corresponding to Equation 13.

The subprogram `linear_solution` inputs two arrays of the same size whose elements are pairwise integers. The output of the subprogram `linear_solution` is an array (D), $2n-2$ elements for input arrays of size n, and whose elements are integers. The subprogram `linear_solution` occurs sequentially after the subprogram `solution_pairs` in the Cosper program. In the Cosper program, `linear_solution` inputs the outputted arrays Ω and A generated by the subprograms `count_a` and `solution_pairs`, respectfully, to generate the array D.

The array D is generated by connecting sequential binary substrings demarked by Ω and A. The process to generate array D begins by reading the elements of Ω and A left to right. From A two sequential elements are selected. For example, $(seed_i, phase_i)$ and $(seed_{i+1}, phase_{i+1})$. From Ω, only one element is selected corresponding to the first of the two elements selected in A, (k_i, j_i).

For our next step, recall from Proposition 2 that the trajectory of a seed integer is not affected by its phase. Here we take this idea one step further by observing that when imposing the Collatz map C on an odd integer the addition of one can be applied solely on the value of the seed alone. Consider Equation 14 below:

$$\mathrm{C}^1(seed_i + phase_i) = (3 \times seed_i + 1) + (3 \times phase_i). \tag{14}$$

Where the parenthesis on the right hand side illustrate that the action of the map can be separated into two parts. Similarly if some seed integer for a trajectory was even the action of the Collatz map can be split into two equivalent pieces for the respective seed and phase. This is seen in Equation 15.

$$\mathrm{C}^1(seed_i + phase_i) = (\frac{seed_i}{2}) + (\frac{phase_i}{2}). \tag{15}$$

Therefore we can split the action of the Collatz map into two distinct parts applied to the seed and phase independently. We will keep the same map C for the seed integer, while for the phase we will use a modified Collatz map ($\tilde{\mathrm{C}}$) that includes no additive constant in the odd case. For this construction, we make the modified Collatz map's action dependent on the Collatz map's action so that the end result of consecutive mappings is equivalent to the Collatz mapping itself. The modified Collatz map appears below in Equation 16.

$$\begin{aligned} &\tilde{\mathrm{C}} : \mathbb{N} \to \mathbb{N} \\ &\tilde{\mathrm{C}}^k(phase) = \begin{cases} 3 \times phase\,, & \mathrm{C}^{k-1}(seed) \equiv 1 \mod 2 \\ \frac{phase}{2}\,, & \mathrm{C}^{k-1}(seed) \equiv 0 \mod 2 \end{cases}. \end{aligned} \tag{16}$$

Before going further, let us consider an example. Let us consider the binary sequence B = 1010101000. The seed integer in this case is 15, and the phase is $64m$. We can tabulate the result of iterate steps of the trajectory below (Figure 9).

Notice here, that the phase has transformed from strictly a power of two to strictly a power of three. This can be understood from the fact that the trajectory was iterated $j_i + k_i$ steps, which ensures that the phase is divided by the same power of two which it was initially defined. Thus, all that remains of that integer is the composition of powers of three which it accumulated from iterate compositions of the $\tilde{\mathrm{C}}$ map. The next step is to connect the trajectories of $seed_i$, $phase_i$ to $seed_{i+1}$, $phase_{i+1}$. To do this we must advance $seed_i$ and $phase_i$ $j_i + k_i$ steps along the C and $\tilde{\mathrm{C}}$ maps, respectfully.

k	$C^k(15)$	$\tilde{C}^k(64)$	$C^k(15+64)$
1	46	192	238
2	23	96	119
3	70	288	358
...	...	...	...
10	20	81	101

Figure 9: Table of trajectories involving a given seed and phase values for C and $\tilde{C}$ with $m = 1$.

This will result in the trajectories of $seed_i$ and $phase_i$, up to a constant u matching the trajectories of $seed_{i+1}$ and $phase_{i+1}$ up to a constant v (see Equation 17).

$$\begin{aligned} C^{j_i+k_i}(seed_i) + \tilde{C}^{j_i+k_i}(phase_i) \times u &= seed_{i+1} + phase_{i+1} \times v, \\ phase_{i+1} \times v - \tilde{C}^{j_i+k_i}(phase_i) \times u &= C^{j_i+k_i}(seed_i) - seed_{i+1}. \end{aligned} \tag{17}$$

To resolve Equation 17, consider first the simpler expression (Equation 18).

$$phase_{i+1}s - \tilde{C}^{j_i+k_i}(phase_i)t = 1. \tag{18}$$

Equation 18 can be solved for by using the Extended Euclidean Algorithm[8]. This algorithm is presented in the following theorem:

Theorem 2.7. *(Bezout's Theorem) Let $d = \gcd(a,b)$ be the greatest common denominator for integers a and b. Then there exists $s,t \in \mathbb{Z}$ such that $d = sa + tb$.[8]*

Having resolved the coefficients s and t from Equation 18 by the Extended Euclidean Algorithm (Theorem 2), we can proceed to further reduce the expression. Let us proceed by reducing Equation 18 to a congruence relationship by taking modulus $phase_{i+1}$. Then multiply both sides by $C^{j_i+k_i}(seed_i) - seed_{i+1}$. This gives us Equation 19:

$$-\tilde{C}^{j_i+k_i}(phase_i)(C^{j_i+k_i}(seed_i) - seed_{i+1})t \equiv (C^{j_i+k_i}(seed_i) - seed_{i+1}) \mod phase_{i+1}. \tag{19}$$

Taking the modulus $phase_{i+1}$ of Equation 17 gives us Equation 20:

$$-\tilde{C}^{j_i+k_i}(phase_i)u \equiv C^{j_i+k_i}(seed_i) - seed_{i+1} \mod phase_{i+1}. \tag{20}$$

Combining Equations 19 and 20, and canceling out $-\tilde{C}^{j_i+k_i}(phase_i)$ on both sides yields the final expression to solve for u in terms of t and the difference of the seed values (Equation 21).

$$u \equiv t(C^{j_i+k_i}(seed_i) - seed_{i+1}) \mod phase_{i+1}. \tag{21}$$

This coefficient u, along with $phase_{i+1}$ are stored into array D. The process proceeds for all sequential pairs until all elements in A (and all but the last element in Ω) have been used. This completes the subprogram `linear_solution`.

The subprogram `combine` takes as an input two arrays and outputs a pair of numbers. The elements of one of the arrays are pairs of numbers, while the elements of the other array are singular numbers. The former array whose elements are pairs of numbers, by hypothesis, possesses two more elements than the latter array. In the Cosper program, the subprogram `combine` occurs after the subprogram `linear_solution`. Additionally, in the Cosper program the inputted arrays are A and D. The outputted pair of numbers are $(seed, phase)$. Observe that there is no subscript on either of these numbers. This is because this outputted pair of numbers generates the family of integers which match the binary sequence B at every step.

Recalling Equation 14, the objective of `linear_solution` was to solve for the coefficients u and v which satisfy each linear expression for the whole binary sequence. The value of u, for every step, was stored in the odd elements of array D. The even elements of the array D are the individual phases for each finite binary subsequence. Recalling also Proposition 2, we know that an integer multiple of the phase for a given finite binary subsequence does not modify the subsequence. Rather integer multiples of the phase modify the binary sequence for elements after the finite subsequence. By solving for Equation 17, we have found a multiple (u) of the phase of $(seed_i, phase_i)$ which at the end of its finite binary subsequence extends into the finite binary subsequence of $(seed_{i+1}, phase_{i+1})$. For clarity, consider the following example:

Let us try to combine two binary sequences B_1 and B_2 into a single binary sequence B_3 (whereas all three binary sequences obey the stipulated rules).

B_1	B_2	B_3
10000	1010100	100001010100

The integer family of B_1 is $\{5+16u | u \in \mathbb{Z}_+\}$, and the integer family of B_2 is $\{7+16v | v \in \mathbb{Z}_+\}$. What we need is that after 5 steps of the C map, the integer family of B_1 must properly reduce to the integer family of B_2. We recall Equation 15 to assist in reducing the integer family in the following Equation 22:

$$C^5(5+16u) = C^5(5) + \tilde{C}^5(16u) = 1+3u. \tag{22}$$

Our requirement for the integer family to be reduced appears in the following Equation 23:

$$\begin{aligned} 1+3u &= 7+16v, \\ 16v-3u &= -6. \end{aligned} \tag{23}$$

Following our previous argument, we begin solving for integers u and v by first solving the reduced expression for integers s and t (see Equation 24).

$$16s + (-3t) = 1. \tag{24}$$

From the Extended Euclidean Algorithm, we find that Equation 24 is solved for $s = 1$ and $t = 5$. From Equation 21 we now can solve for u (Equation 25).

$$\begin{aligned} u &\equiv t(\mathrm{C}^{j_i+k_i}(seed_i) - seed_{i+1}) \mod phase_{i+1} \\ &\equiv 5(1-7) \mod 16 \\ &\equiv -30 \mod 16 \\ &\equiv 2 \mod 16. \end{aligned} \tag{25}$$

This value of u is stored into an odd element of D while $phase_{i+1}$ is stored in the consecutive even element of D. For our example D = [2,16], and we are done with this step. What we will find using Equations 25 and 26 is that the desired family of integers to generate B_3 is $\{37 + 256m | m \in \mathbb{Z}_+\}$.

In this example we found that for $m = 2$ that the integer family of 5 contains the desired finite binary subsequence generated by 7 for the first seven steps. From Theorem 2, we know we can always do this for any pairwise finite binary subsequences. To connect multiple finite binary subsequences in order, we merely compose each coefficient u (the odd elements of D) with the respective phase (even elements of D) to preserve it in each subsequent integer family. The subprogram `combine` merely follows through with such a composition using a basic recursion relation which can be expressed in the following equation (Equation 26). Whereas in this case the coefficient n in the upper part of the summand is half the length of array D ($n = \frac{|\mathrm{D}|}{2}$). The interior of the parenthesis in Equation 23 can be read as a sum of products: $\mathrm{D}[1] + \mathrm{D}[3]\mathrm{D}[2] + \mathrm{D}[5]\mathrm{D}[2]\mathrm{D}[4] + \ldots$. Its presentation below is the compact way of stating this relationship.

$$seed = \mathrm{A}[1][1] + \mathrm{A}[1][2]\Big(\sum_{i=1}^{n} \mathrm{D}[2i-1] \prod_{j=1}^{i-1} \mathrm{D}[2j]\Big). \tag{26}$$

The complete phase is found by taking the power of two which represents the number of zeros in the desired binary sequence (Equation 27):

$$phase = 2^{j+k}. \tag{27}$$

This pair of numbers $(seed, phase)$ is returned and is the complete solution for the binary sequence B. This completes the subprogram `combine`, as well as the Cosper program.

For example, consider the following binary sequence:

$$\begin{aligned}
\mathrm{B} &= 10101001001001000010101 00\\
\Omega &= [[3,1],[1,1],[1,1],[1,3],[3,1]]\\
\mathrm{A} &= [[7,16],[1,4],[1,4],[5,16],[7,16]]\\
\mathrm{D} &= [0,4,3,4,13,16,11,16]
\end{aligned}$$

From array D, we can generate the seed integer which will match the binary sequence B using Equation 26:

$$\begin{aligned}
\text{seed} &= \mathrm{A}[1][1] + \mathrm{A}[1][2] \times (\mathrm{D}[1] + \mathrm{D}[3] \times \mathrm{D}[2] + \mathrm{D}[5] \times \mathrm{D}[4] \times \mathrm{D}[2] + \mathrm{D}[7] \times \mathrm{D}[6] \times \mathrm{D}[4] \times \mathrm{D}[2])\\
&= 7 + 16 \times (0 + 3 \times 4 + 13 \times 4 \times 4 + 11 \times 16 \times 4 \times 4)\\
&= 7 + 16 \times 0 + 16 \times 4 \times 3 + 16 \times 4 \times 4 \times 13 + 16 \times 4 \times 4 \times 16 \times 11\\
&= 7 + 0 + 192 + 3328 + 45056\\
&= 48583.
\end{aligned}$$

Additionally the phase can be solved for by counting the number of zeros in B:

$$phase = 2^{16} = 65536.$$

Hence the family of integers which matches B can be expressed in the following manner:

$$\mathrm{B}_m = \{48583 + 65536m | m \in \mathbb{Z}_+\}.$$

3 Experiments

3.1 Periodic Orbits of the $3x+k$ map

In this section we take a step back from our study of the $3x+1$ Problem to an immediate generalization. That is, lifting the Collatz Function from $3x+1$ to $3x+k$, for some odd positive integer k. We introduce such a function formally in the following definition.

Definition 3.1. A Collatz-like $3x+k$ function is a function mapping the positive integers to the positive integers of similar manner to the Collatz (C) map. Different though is the additive constant 1 is substituted by k for an odd positive integer k. This substitution is denoted by the subscript k of the form C_k. This is presented in Equation 28.

$$\begin{aligned}
&\mathrm{C}_k : \mathbb{N} \to \mathbb{N}\\
&\mathrm{C}_k(n) = \begin{cases} 3 \times n + k, n \equiv 1 \mod 2 \\ \frac{n}{2}, n \equiv 0 \mod 2 \end{cases}.
\end{aligned} \tag{28}$$

In this context we can revisit the original conjecture regarding the $3x+1$ Problem and resolve the statement via the following proposition.

Proposition 3.2. *For a given binary sequence B which follows the three stipulated rules, let* ρ *be the number of zeros, and* ν *be the number of ones. Let* m_j *be the number of zeros between the* $(j-1)$*th and the jth one. Then the* $3x+k$ *map has a periodic orbit if either:*

1. $2^\rho - 3^\nu$ *divides* k,
2. $2^\rho - 3^\nu$ *divides* $\left(3^{\nu-1} + \sum_{r=1}^{\nu-1} 3^{\nu-1-r} 2^{\sum_{j=1}^{r} m_j}\right)$.

Under the conditions that $\rho > \frac{ln(3)}{ln(2)}\nu$ *and* $\sum_{j=1}^{\nu} m_j = \rho$.

Proof. Let us prove this proposition by construction. Suppose we are given a k in a Collatz-like $3x+k$ function such that a seed integer n has a periodic orbit of $\rho+\nu$ steps (Equation 29).

$$n = C_k^{\rho+\nu}(n). \tag{29}$$

Suppose first that there exist no odd positive integers in the periodic orbit. Then by Equation 28, the orbit of n returns to its original value after being divided by $\rho+\nu$ powers of two. This is impossible unless n is zero, leading to a contradiction. Therefore there is at least one odd positive integer in the orbit of n. For convenience, let us assume that n is odd. Else, advance the orbit to the first odd positive integer and rewrite the equality from Equation 29 using that value.

Applying the map, C_k once yields the following expression:

$$n = C_k^{\rho+\nu-1}\big(3n+k\big).$$

Both the terms $3n$ and k are odd. Therefore their sum is even. We can divide by m_1 zeros yielding the following expression:

$$n = C_k^{\rho-m_1+\nu-1}\big(\frac{3n+k}{2^{m_1}}\big).$$

By hypothesis this expression is again odd. We can apply the map C_k one more step.

$$n = C_k^{\rho-m_1+\nu-2}\big(3\big(\frac{3n+k}{2^{m_1}}\big)+k\big).$$

Again by hypothesis, this expression is even. This allows us to divide by m_2 powers of 2. Following this, we can rearrange the equation into a form which presents the pattern more clearly:

$$n = C_k^{\rho-m_1-m_2+\nu-2}\Big(\frac{3\frac{3n+k}{2^{m_1}}+k}{2^{m_2}}\Big) = C_k^{\rho-m_1-m_2+\nu-2}\Big(\big(\frac{3^2 n}{2^{m_1+m_2}} + k\big(\frac{3}{2^{m_1+m_2}} + \frac{1}{2^{m_2}}\big)\Big).$$

Following the process through the entire periodic orbit of n yields the following expression:

$$n = \frac{3^{\nu} n}{2^{\rho}} + k\Big(\frac{3^{\nu-1}}{2^{\rho}} + \frac{3^{\nu-2}}{2^{\rho-m_1}} + \frac{3^{\nu-3}}{2^{\rho-m_1-m_2}} + \ldots + \frac{1}{2^{m_{\nu}}}\Big).$$

Now combining both n-terms on the left-hand side, multiply the entire expression by 2^{ρ}, and finally dividing the entire expression by $2^{\rho} - 3^{\nu}$ yields Equation 30.

$$n = \frac{k\big(3^{\nu-1} + \sum_{r=1}^{\nu-1} 3^{\nu-1-r} 2^{\sum_{j=1}^{r} m_j}\big)}{2^{\rho} - 3^{\nu}}. \tag{30}$$

From here, it is clear n is only an integer if $2^{\rho} - 3^{\nu}$ divides k or divides the quantity in parenthesis. The first condition, that $\rho > \frac{ln(3)}{ln(2)}\nu$ prevents the denominator from either becoming zero (if inequality is expressed as equality) or from becoming negative (which is outside of the hypothesized domain). The second condition is met by the construction. Hence, by construction, we have created a periodic orbit for n under the $3x + k$ map. □

Proposition 3 allows us to create two new definitions. We can separate the periodic orbits of any $3x + k$ map into two categories depending upon which term $2^{\rho} - 3^{\nu}$ divides. That is, both generate periodic orbits, but under completely different circumstances.

Definition 3.3. k-dependent cycles are periodic orbits which Condition 1 of Proposition 3 is satisfied. The periodic orbit exists exclusively for the chosen k, for a given seed integer n.

Definition 3.4. k-independent cycles are periodic orbits which Condition 2 of Proposition 3 is satisfied. The periodic orbit exists regardless of k, for a given seed integer n.

For example, let us create a k-dependent cycle of 8 steps. For convenience, let this periodic orbit correspond to the following binary sequence: 10101000. There are 5 zeros ($\rho = 5$) and 3 ones ($\nu = 3$). We have chosen to order the ones and zeros such that the parity alternates until all the ones have been exhausted. At that step, the sequence concludes with an extra 2 zeros. From Condition 1 of Proposition 3, the simplest choice of k would be setting $k = 2^5 - 3^3 = 5$. In this context, every single m_j less than m_2 is one. Therefore we can express n in the form:

$$\begin{aligned} n &= 3^2 + \sum_{r=1}^{2} 3^{2-r} 2^{\sum_{j=1}^{r} m_j} \\ &= 3^2 + \sum_{r=1}^{2} 3^{2-r} 2^{\sum_{j=1}^{r} 1} \\ &= 3^2 + 3^1 2^1 + 2^2 \\ &= 9 + 6 + 4 = 19. \end{aligned}$$

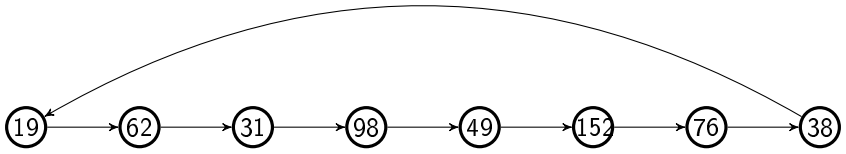

Figure 10: Constructed cycle of 3x+5 map with seed value of 19.

This cycle appears as follows in Figure 10 :

Note: It is assumed here that the only size of the orbit is $\rho + \nu$. Let us make this assumption clear with the following proposition (Proposition 4).

Proposition 3.5. *A given finite binary sequence B following the three stipulated rules uniquely corresponds to a finite integer cycle of a* $3x + k$ *map up to a change of position in that cycle.*

Proof. Let us begin by clarifying what is meant by a change in position in the cycle. We can tabulate the example cycle of the $3x+5$ map. Below is its seed value of 19, its binary sequence, its summation, and all subsequent integers in the same cycle as seen in Figure 11.

19	**1**0101000	$3^2+3^1\times 2^1+2^2$
62	0101000**1**	$2^1\times(3^2+3^1\times 2^1+2^4)$
31	1010001**0**	$3^2+3^1\times 2^1+2^4$
98	0100**1**101	$2^1\times(3^2+3^1\times 2^3+2^4)$
49	1000**1**010	$3^2+3^1\times 2^3+2^4$
152	000**1**0101	$2^3\times(3^2+3^1\times 2^1+2^2)$
76	00**1**01010	$2^2\times(3^2+3^1\times 2^1+2^2)$
38	0**1**010100	$2^1\times(3^2+3^1\times 2^1+2^2)$
19	**1**0101000	$3^2+3^1\times 2^1+2^2$

Figure 11: Tabulated cycle of 3x+5 map with seed value of 19 with binary and summation representations.

Notice that we are using the same formula provided in Condition 1 of Proposition 3 for column 3, with the exception that we are bending the rule slightly for even integers in the cycle. In fact, these numbers can be assembled using Condition 1 strictly by initially removing the precursory zeros, solving for the respective odd integer, then multiplying by the correct power of two which matches the number of zeros omitted. The single leading 1 is written in bold. This is not to indicate that this bit is 19, but rather that when the binary sequence of the cycle is written with that bit in front, it will generate our seed value of 19.

Observe that every forward step of the $3x+5$ map shifts the bits one value to the left. This shift will continue until the trajectory completes its cycle and returns to the original seed value. Moreover, this would occur for any other choice seed value anywhere in the cycle. That is, the cycle must be generated by any value in the cycle along with the $3x+5$ map. To that degree, the binary sequence is not unique with respect to a left shift of a given binary sequence corresponding to a seed value within the cycle. However, as all the odd integers can be manifested using Condition 1 (and the even integers by stripping the precursory zeros and multiplying by the correct power of 2) we can still claim that the cycle is $\rho+\nu$ steps.

Now let us consider the possibility of creating the same cycle but with a number of steps not equal to $\rho+\nu$. We begin by considering what a counterexample must satisfy:

1. One seed number must be exactly the same in both cycles.

2. The equation:

$$\begin{cases} 2^a - 3^b = k \\ 2^c - 3^d = k, \end{cases} \tag{31}$$

must produce the same value of k, where by hypothesis $c+d > a+b$.

In fact, it is reasonable to hypothesize condition (2). Indeed, there are many examples of choice a, b, c, and d which, in combination, produce the same k. For example, $2^3-3^1 = 2^5-3^3 = 5$. However, notice immediately in this example there is no arrangement of three zeros and a one which under Condition 1 of Proposition 3 will yield 19 which was our example seed number. Independently (2) can be created using any normal means.

Considering now condition (1), there are also many ways which a longer binary sequence can produce an identical seed number n. The simplest method is appending zeros at the end of the sequence. This is because m_ν is not used in the summation according to Condition 1 of Proposition 3. As long as only zeros are appended at the end of the sequence, the seed value does not change.

The first attempt at creating a counterexample would be combining these two observations together. However, from what we observed about how bits in a binary sequence are left shifted after each iterate computation of the $3x+k$ map, it is an immediate consequence that the next value in the trajectory will not match the binary sequence which appears subsequently in the trajectory of the given cycle. We observe this, as we presume in this counterexample that the new binary sequence must return to the same seed value. Left shift the sequence so that the appended zeros are at the front of the binary sequence (in our example case this would be the number 152). Notice here that there are m_ν power of twos between that step, and the seed number. As this power of two is by hypothesis a greater power of two than the original cycle then the integer at this step does not appear in the original cycle. Thus it is a different cycle.

A harder question would be to ask whether there is a means of arranging a longer binary sequence to recreate the same seed number apart from the method stated above. The simplest answer that can be given here is that each binary digit in the sequence is representative of a step in the cycle. If in fact we found such a arrangement we will also find that cycle, by necessity, must consist of that many more unique integers. No repeat integers can exist, lest the cycle return to the seed value before its completion, or the $3x+k$ map becomes an association, not a function.

□

We will return to the periodic structure of this k-dependent cycle later on. For now though this demonstrates that the construction specified by Proposition 3 and Equation 30 effectively generates a cycle. Again to reiterate, by Proposition 3, this cycle exists for our choice of k. Plugging our constructed n into another k would not generate the same cycle.

Let us now briefly discuss k-independent cycles. We know from Proposition 3 that such periodic orbits exist for any odd positive k. Most notably is the case where $k=1$, is the form of the original conjecture. This observation allows us to rewrite the conjecture in the following form (Proposition 5):

Proposition 3.6. *The Collatz Conjecture is true if:*

$$2^{\rho}-3^{\nu} \nmid 3^{\nu-1}+\sum_{r=1}^{\nu-1} 3^{\nu-1-r}2^{\sum_{j=1}^{r} m_j}, \tag{32}$$

for all binary sequences B following the stipulated rules which do not corresponding to the trivial cycle 1-4-2-1...

For example, we can consider the trivial cycle represented by all binary sequences B of form $\overline{100}$. This is simply because the trivial cycle can be repeated multiple times, and still represent the same cycle of itself. In its simplest form there are two zeros and one, one. This corresponds to $2^2-3^1=1$. The summation includes only one term, $3^0=1$. Consequently as the denominator divides the summation identically, the seed integer is just our choice of k! We can see why by considering that for the Collatz-like function C_k, $k\to 4k\to 2k\to k$. As our k was arbitrary this is true for all odd positive k. Hence, the trivial cycle is k-invariant, as specified by the definition. If any other such cycle could be found, it would render the Collatz Conjecture to be false. Consequently Proposition 5 has reintroduced the conjecture in the form of binary sequences.

3.2 Imaging of k-dependent cycles

It may be quite simple to write down a binary sequence which corresponds to a k-dependent cycle. What is not so simple is finding a means to express the consecutive integers which correspond to the cycle. In the case of an example periodic cycle with one

hundred steps, it would fill up a good portion of the page to write all the numbers down. Instead, let us consider a means of drawing the orbit so that by looking at the picture we know the behavior of the cycle. In this context, we present a program cosper_walk.

The program cosper_walk is a program which generates fully customizable pictorial representations of k-dependent cycles. In its current state, it is restricted to cycles which the value of k is not divisible by three. This is intended to be improved in the future, but is still quite useful as is. To understand its inputs, we will address each independently:

1. The first input is a natural number that is divisible by three. This number corresponds to the number of 'orthogonal directions' the drawing may possess. These 'orthogonal directions' correspond to the modularity of the inputted number. On each inductive step of C_k, the direction of the drawing changes depending upon the arrived modularity. For example, by setting a equal to nine, the numbers 1,2,4,5,7,8 are not divisible by three. They represent the possible modularity which could occur in the co-domain of C_k. They are evenly spaced (see Figure 12) so that on each inductive step of C_k the current modularity can be easily understood.

2. The second input is a scale factor. For each inductive step, the drawing follows a rotation based on the modularity but also advances so that the change is visible. A small scale corresponds to a small drawing and vice versa.

3. The third input is our value of k, which we suggest using the exponent difference $(2^\rho - 3^\nu)$ found in Equation 30.

4. The forth input is our seed integer n, we again suggest using the summation found in Equation 30.

5. The fifth and sixth inputs correspond to advanced features regarding changing the angles for each direction as well as generating an audio signal corresponding to the modularities on each step of C_k.

Having inputted all the desired values, the program will follow into an infinite loop generating the cycle as a computer image. Some example images are presented below in Figure 13. The labeling of each diagram follows a basic procedure: (modularity, scale, k, n, angle modification). Using cosper_walk with the specified inputs will generate each image identically.

These examples were chosen to illustrate an interesting relationship between binary sequences and modularities. We find that the more "restrictive" (sequences with minimal amount of zeros between each one) the binary sequence is, the more "restrictive" the modularity of the integers are. Consider for example the bottom left image in Figure 13. It is generated from a sequence of the form $(10)^{47} + (0)^{28}$. The angles were chosen so that when the integers in the trajectory are modulo one or two with respect to nine, then the

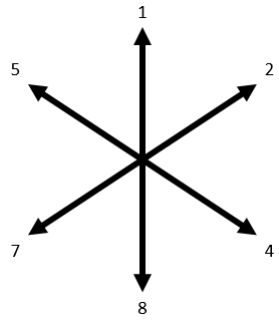

Figure 12: The set of orthogonal directions, not divisible by three, mod 9.

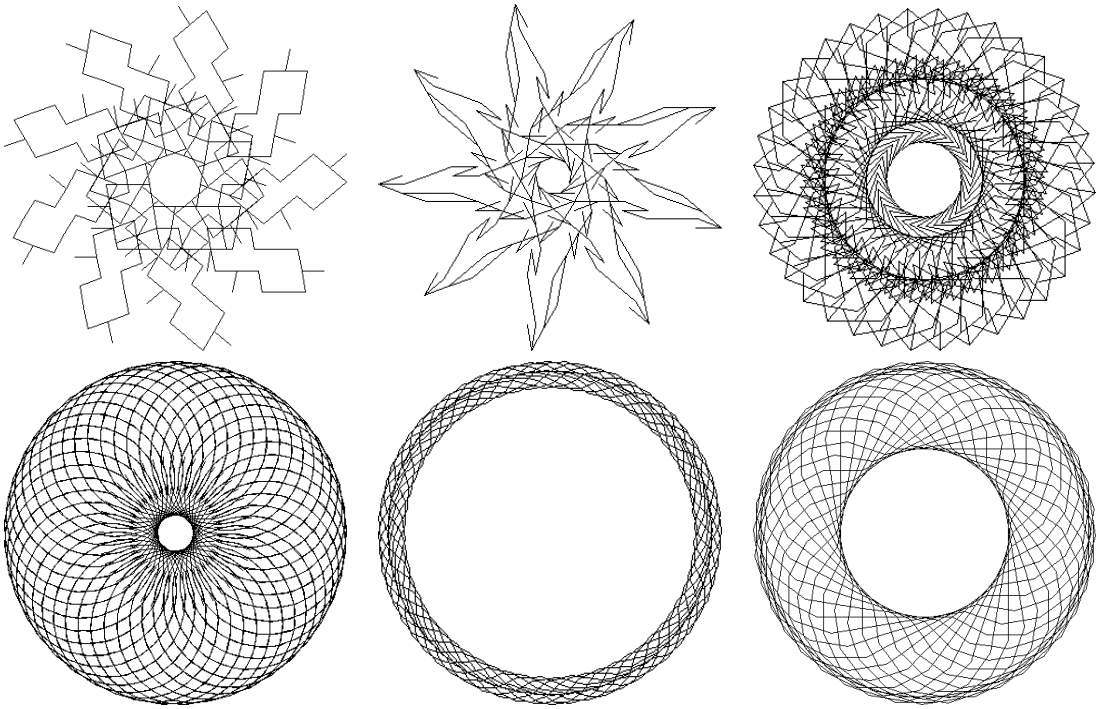

Figure 13: From top left to bottom right:
9,25,68719470175,275060245, [0,20,160,0,20,-200]
9,25,68719470175,275060245, [0,-40,40,0,-40,40]
9,25,68719470175,275060245, [0,-10,10,0,-10,10]
9,25,111901175039996584217 81,26588814218220014932459, [346,-60,74,-14,-60,-286]
6,7,45853,87302, [338,-90,-158,-270]
6,25,45853,25411, [333,-90,-153,-270]

angle between them is quite small. This creates the large circular shape that is rotated about its center. Once the cycle crosses the repeated zeros, the modularity oscillates

between different values modulo nine.

The other five examples share a similar premise where specific repeated modularities presented themselves and whose character was exaggerated by changing the angle of its direction when drawn. For now this is merely a qualitative observation, and does not reflect any rigorous statement at this time. We can observe clear differences in the structure of the image not only by changing the k-dependent cycle, but by changing the angles between each orthogonal direction. As far as the authors know, k-dependent cycle imaging is an unexplored topic that is left open for further inquiry.

4 Conclusion

The Collatz Conjecture remains to be an unsolved problem in Mathematics. It has outwitted every modern method of proof, making its simple appearance extraordinarily misleading. In this paper we presented the history, modern progress, Cosper Program, as well as some applications of k-dependent cycles. It is a hope that the ideas presented here will lead to greater insight into the problem, especially Proposition 5.

5 Acknowledgments

I would first like to thank Mr. Alexander Bairrington who introduced me to this problem over eleven years ago. Although the rainy Saturday afternoons at the family's kitchen table are long gone, my curiosity for this problem has remained vigilant ever since.

I would also like to thank Dr. David Cherney for his patience and support in transforming my cryptic nomenclature into a presentable piece of mathematical literature. Without his help, this paper would have never left my notebooks.

Dr. John Hunter deserves many thanks for his practical advice and support throughout this project. His patience served to refine the project into its current form.

Many thanks are given to Mr. Aaron Okano for his insightful knowledge and programming finesse. The computational power of the Cosper program would have never left infancy without his rigorous support.

Lastly, I would like to thank Dr. Jesus De Loera for his continual mentorship in and out of University. He has given me a deep love of Mathematics that I know I will carry throughout my career.

6 Appendix

6.1 Introduction

The target audience for this supplementary material for the Cosper Program is undergraduate students, mathematical hobbyists, and mathematical researchers. General

mathematical literacy is expected, with some knowledge of key terms presented in the original paper. The purpose of this supplementary material is to describe the process, limits of computation, and standard operation procedure of the Cosper Program. Emphasis is placed on the ternary attribute so that the reader (by the end of this document) would be able to operate the Cosper Program without any difficulty. The Cosper Program is scripted in python. The python language was used for its ease of access of interchanging number types, and for the simplicity of the Tkinter graphical user interface package. There are no anticipated revisions to the program in the immediate future. Any comments, questions, or concerns can be forwarded to i_is_confused@yahoo.com and will be responded to at the soonest convenience.

6.2 Cosper Program Process

The Cosper Program is broken up into two python files. One is called cosper_program.py and the other cosper_computation.py. cosper_program.py represents the GUI script, while cosper_computation.py represents the computational process. The user input is taken from cosper_program, although if one were to open up the script in cosper_computation.py an input could be written directly.

Once the user input has been taken from the GUI, cosper_computation is called as a subprogram to engage the computational process. The output is placed both on the DOS partition of the GUI as well as stored in an external file named cosper_output.txt, which will appear in the same file location as cosper_program.py.

Referencing the original paper, cosper_computation.py can be broken up into the following subprograms: count_a, solution_pairs, linear_solution, and combine. Each subprogram plays a role to convert the provided user input of the form of a binary sequence into a family of integer solutions under the Collatz-map C. Count_a takes the user input from cosper_program.py and counts the number of consecutive (10) and (0) elements in the user input. This data is carried to solution_pairs, where an application of Euler's Theorem is applied to generate the multitude of integer families which satisfy each unique partition of the binary sequence. Linear_solution uses an application of the Extended Euclidean Algorithm to find a list of integers which combine each subsequent integer family. Combine takes these integer families, with the list of integers generated in linear_solution to create a single integer family which matches the original user input. Further detail regarding the mathematical theory of the program can be referenced in the original paper.

6.3 Limits of Cosper Computation

Using randomized binary sequence inputs, we have found that a standard computer with python 3.3 can handle binary sequences up to 100,000 elements long. For sequences of that length, a program run-time in five minutes is expected. Under this circumstance

linear_solution becomes a bottleneck process, as the list of integers which would be used to generate the complete integer family becomes extremely long (see Equation 26 of the paper). This is left as an open question to find a means to make this subprogram run faster.

6.4 Standard Operation Procedure of the Cosper Program

A working version of python is required to run the program. Such a version must have access to the libraries: Tkinter, cprofile, turtle, random. These usually come with any downloadable version of Python, but are free to download separately if necessary.

There are three files which be in the same file location for the program to run: cosper_computation.py, cosper_program.py, cosper.gif. In case any of these files are missing the program will not be operational.

To begin, run the cosper_program.py program. It will generate a graphical user interface GUI appearing as Figure 14:

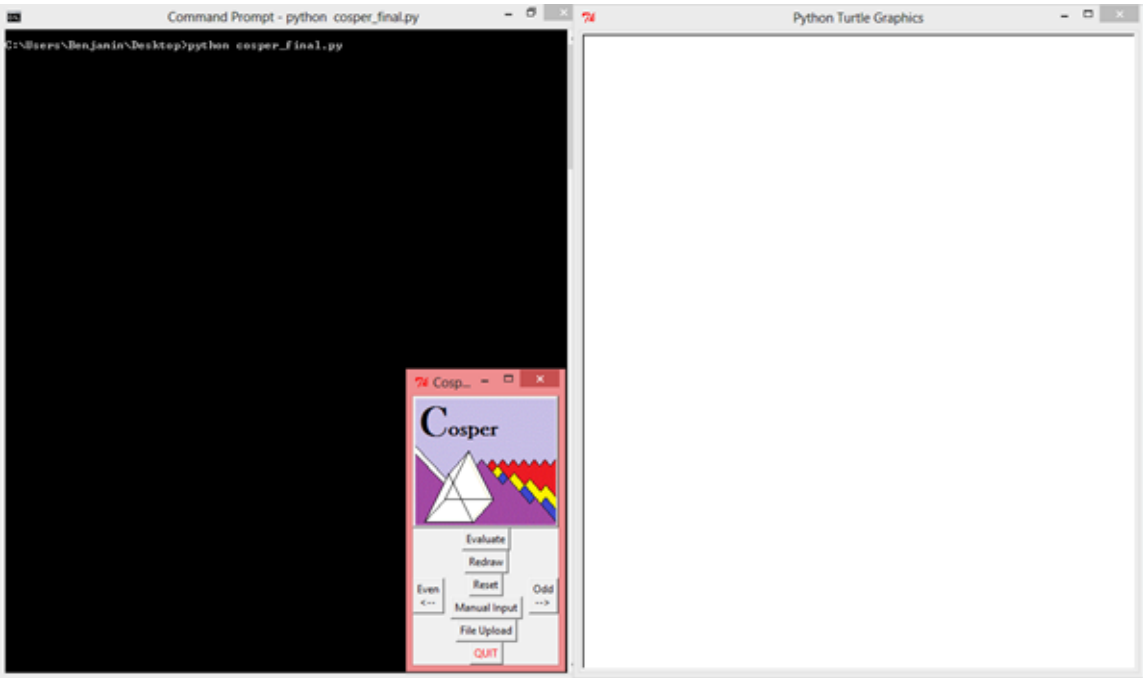

Figure 14: The GUI interface of cosper

There are three window panes in the Cosper GUI interface: the DOS window (on left), the diagram screen (right), and the Cosper remote (bottom center).

The DOS serves three purposes: to display the binary user input, show the computational process, and to output the integer family.

The diagram screen creates a visual representation of the user input with one of two strokes: up-right representing an odd number (or one) and down-right for an even number (or zero). This is mean to give a visual image to the binary string, which many not be easy to visualize by itself.

The Cosper remote is the pane which all user input is taken from. It consists of eight buttons which are described below:

Evaluate – calls cosper_computation.py for a given user input generating a solution both as a text file cosper_output.txt and on the DOS screen.

Redraw – clears the diagram screen and redraws the diagram representing the user input.

Reset – clears the diagram screen as well as the user input.

Manual Input – creates a small dialogue box which the user can input a binary string (no spaces, commas, or delimiters) directly. The input is not accepted unless the user presses the corresponding 'Ok' button that appears with the dialogue box. When the 'Ok' button is pressed the previous input is deleted, the new input is accepted and is drawn on the diagram screen.

File Upload – creates a small dialogue box in which the user can input an external file which will be read (no spaces, commas, or delimiters). The input file name needs to include the file extension (such as .txt). The input is not accepted until the user presses the 'Upload File' button that appears in the dialogue box. The external file input is displayed on the DOS screen but is not displayed on the diagram screen unless the user presses the 'redraw' button. Like the 'manual input' button, the previous input is deleted before creating the new input from the file. For large binary sequences, it is not recommended to press the 'redraw' button.

Quit – Closes the Cosper remote. The program does not halt unless the diagram screen is also closed as well.

Even – Allows the user to input (either initially, or as an extension of previous input) an extra (0) at the end of a binary sequence. This appended (0) is also drawn on the diagram screen.

Odd – Allows the user to input (either initially, or as an extension of a previous input) an extra (10) at the end of a binary sequence. This represents both the odd, and mandatory even number which occur in the new binary sequence input. This mandatory addition of a zero to the input prevents any errors that may occur in cosper_computation.py under the case the input binary sequence includes a (11). This (10) appears together on the diagram screen as an up-right, and down-right strokes, respectively.

6.5 Example Operation

What follows is a demo operation of the Cosper Program. In this demo, the binary sequence 101000000101000 is manually inputted, and then 100 is added at the end using the 'Even' and 'Odd' buttons. After the input is complete, the binary sequence is evaluated, and the program is shut down.

1. Check that cosper_program.py, cosper_computation.py, and cosper.gif are all in the same file location.
2. Run cosper_program.py.
3. From the cosper remote, select the 'manual input' button. A dialogue box with an 'Ok' button will appear below.
4. Type the binary sequence 101000000101000 into the dialogue box, and click 'Ok'. The binary sequence will be drawn on the cosper drawing screen, and appear in the DOS screen as an array.
5. After the sequence has been drawn, select the 'Odd' then 'Even' buttons to append 100 to the binary sequence input. After each consecutive button push, the cosper drawing screen will update with each input.
6. Press the 'Evaluate' button.
7. Looking at the DOS screen, the bottom line (below the row of stars) will have the integer family of the form: $1635 + 8192 * m$.
8. Press the 'Quit' button on the cosper remote, followed by closing the cosper drawing screen. The DOS screen will return to a nonoperative state.

6.6 Known Errors

If one selects the 'Manual Input' or 'File Upload' button twice, two dialogue boxes will appear. As the program will expect input in both dialogue boxes simultaneously, the input may not necessarily work. The simplest solution is to restart the program.

The dialogue box corresponding to the 'File Upload' button requires that the file name also include the extension. If there is no extension, then the file will fail to upload.

6.7 Remarks

We sincerely hope that this brief introduction is sufficient to describe the Cosper Program in a manner which is both concise, and informative. We also hope that this tool will be useful to many of whom find this problem as intriguing as we do.

References

[1] J.F. Alves, M.M. Graça, M.E.S. Dias, and J. S. Ramos. A linear algebra approach to the conjecture of Collatz. *Linear algebra and its applications*, 394:277–289, 2005.

[2] L. Collatz. On the motivation and origin of the $(3n+1)$ Problem (Chinese). *J. Qufu Normal University, Natural Science Edition [Qufu shi fan da xue xue bao]*, 12(3):9–11, 1986.

[3] L. Colussi. The convergence classes of Collatz function. *Theoretical Computer Science*, 412(39):5409–5419, 2011.

[4] H.S.M. Coxeter. Cyclic sequences and frieze patterns. In *The Ultimate Challenge: The* $3x+1$ *Problem*, pages 211–217. Amer. Math. Soc., 2010.

[5] J.M. Dolan, A.F. Gilman, and S. Manickam. A generalization of Everett's result on the Collatz $3x+1$ problem. *Advances in Applied Mathematics*, 8(4):405–409, 1987.

[6] C.J. Everett. Iteration of the number-theoretic function $f(2n) = n$, $f(2n+1) = 3n+2$. *Advances in Mathematics*, 25(1):42–45, 1977.

[7] M. Gardner. Mathematical games. *Scientific American*, 226(6):114–121, 1972.

[8] S. P. Glasby. Extended Euclid's Algorithm via backward recurrence relations. *Mathematics Magazine*, 72(3):228–230, 1999.

[9] H. Hasse. Unsolved problems in elementary number theory. *Lectures at U. Maine (Orono)*, Spring 1975.

[10] J.C. Lagarias. The $3x+1$ Problem and its generalizations. *American Mathematical Monthly*, pages 3–23, 1985.

[11] J.C. Lagarias. The $3x+1$ Problem: An overview. In *The Ultimate Challenge: The 3x+ 1 Problem*, pages 3–29. Amer. Math. Soc., 2010.

[12] P. Michel and M. Margenstern. Generalized $3x+1$ functions and the theory of computation. In *The Ultimate Challenge: The 3x+1 Problem*, pages 105–130. Amer. Math. Soc., 2010.

[13] G. Nürnberger. Günter Meinardus (1926–2007). *Journal of Approximation Theory*, 162(1):1–5, 2010.

[14] T. Oliveira e Silva. Empirical verification of the $3x+1$ and related conjectures. In *The Ultimate Challenge: The 3x+ 1 Problem*, pages 189–207. Amer. Math. Soc., 2010.

[15] G. Opfer. An analytic approach to the Collatz $3n+1$ Problem. *Hamburger Beiträge zur Angewandten Mathematik*, vol. 9, 2011.

[16] J. P. Van Bendegem. The Collatz Conjecture. A case study in mathematical problem solving. *Logic and Logical Philosophy*, 14(1):7–23, 2005.

[17] J. J. Watkins. *Number Theory A Historical Approach*. Princeton University Press, Princeton, NJ, 2014.

[18] J.R. Whiteman. In memoriam: Lothar Collatz. *International Journal for Numerical Methods in Engineering*, 31(8):1475–1476, 1991.

Benjamin Bairrington
University of California, Davis
i_is_confused@yahoo.com

Aaron Okano
University of California, Davis