The proof of a long standing conjecture of Erdös on divisibility of the central binomial coefficients by 105

Ernest S. Croot, Hamed Mousavi, and Maxie D. Schmidt

October 4, 2018

Abstract

A long standing open question in combinatorial number theory due to Erdös is whether there are infinitely-many natural numbers n for which $\gcd(\binom{2n}{n}, p)$ is simultaneously equal to 1 for all $p \in \{3, 5, 7\}$. In the 1990's, Ronald Graham offered a prize for a proof of the infinitude of such n with a prime bounty totaling \$1000. Much has been made of this open question, though to date no existing proof of this result, or counterexample thereof, is known. We present and prove an exhaustive method for determining by computerized search that there must be infinitely-many such n.

While this problem is seemingly an infinite problem, we prove the correctness of an algorithm which allows us to enumerate by brute force search a so-called "base" set of solutions, and then extend the existence of these base cases to show the infinitude of such n in the problem. Our algorithm requires at most roughly 3^{21} integer or floating point operations. With the advent of modern multicore processors the solution to Erdös's conjecture is finally attainable using our algorithmic argument. Suggested generalizations of this result based on a modification of our proof is possible.

1 Introduction

1.1 A conjecture of Erdös and Graham

Erdös conjectured in the 19xx's that the set of natural numbers

$$\begin{aligned}\mathcal{B}_{3,5,7} &:= \left\{n \geq 1 : \gcd\left\{\binom{2n}{n}, 105\right\} = 1\right\} \\ &= \{1, 10, 756, 757, 3160, 3186, 3187, 3250, 7560, 7561, 7651, 20007\ldots\},\end{aligned}$$

is infinite. The statement is equivalent to showing that there are infinitely many positive integers n such that the base-p expansion of n, denoted $(n)_p := d_{1,p}^{(n)} d_{2,p}^{(n)} d_{3,p}^{(n)} \cdots$, simultaneously satisfies $d_{i,3}^{(n)} \in \{0, 1\}$, $d_{i,5}^{(n)} \in \{0, 1, 2\}$, and $d_{i,7}^{(n)} \in \{0, 1, 2, 3\}$. This equivalent statement of Erdös's conjecture follows as a well-known consequence of Kummer's theorem on the p-divisibility of the binomial coefficients. Practically speaking, for fixed $n \geq 1$ we can always assume that the base-p expansion $(n)_p$ has only finitely-many non-trailing zeros.

1.2 A basic heuristic of expectation on the problem

If we choose an integer $n \leq x$ at random, we can ask what the probability is that all its digits in those bases are the possibilities listed. We can't calculate this probability exactly – if we did, then we could already solve the problem – but we can give a plausible heuristic as to its value, by assuming the bases act independently. For x large, the probability that all the digits are 0 or 1 in base-3 is about $(2/3)^{\log(x)/\log(3)}$, since there are at most about $\log(x)/\log(3)$ digits base-3 among all $n \leq x$. For base-5 the probability estimate is $(3/5)^{\log(x)/\log(5)}$, and for base-7 it is $(4/7)^{\log(x)/\log(7)}$. So, the number of integers $n \leq x$ that hit all three requirements, in all three bases, assuming indepedence, is

$$x(2/3)^{\log(x)/\log(3)}(3/5)^{\log(x)/\log(5)}(4/7)^{\log(x)/\log(7)}$$

Taking logs, this is

$$\log(x)\left(1+\log(2/3)/\log(3)+\log(3/5)/\log(5)+\log(4/7)/\log(7)\right) \approx 0.02595\log x.$$

So, there should be about $x^{0.02595}$ integers $n \leq x$ such that $\binom{2n}{n}$ is coprime to 105 [3, cf. §4].

1.3 Foundations of the contruction of our new algorithm

We use the base-$3,5,7$ characterization of the natural numbers $n \geq 1$ to construct a finite algorithmic search which enumerates a "base" set of integers n. This so-called working base set can then be extended to an infinite set of integers with our key property. This algorithm is based purely on an initital working hypothesis, which we prove below, using truncated base-p expansions ($p = 3,5,7$) of real numbers within some bounded set. In particular, we prove the following result:

Theorem 1 (Working Hypothesis). *There exist integers $1 \leq B_1 < B_2$ such that for every set of four real numbers $\alpha,\beta,\gamma,\delta$ satisfying*

$$0 < \beta,\ \delta < 3^{-B_1};\ 1 \leq \alpha < u_\alpha < \infty;\ 1 \leq \gamma < u_\gamma < \infty,$$

we have that there exist distinct integers

$$1 \leq x_1 < x_2 < \cdots < x_t \leq B_2,$$

such that if we write the number

$$\alpha\left(\frac{1}{3^{x_1}}+\frac{1}{3^{x_2}}+\cdots+\frac{1}{3^{x_t}}\right)+\beta$$

in its base-5 expansion,

$$\frac{d_1}{5}+\frac{d_2}{5^2}+\cdots+\frac{d_u}{5^u}+\varepsilon_1,\ u=\lceil x_t\log(3)/\log 5\rceil, \tag{1}$$

then

$$d_1,d_2,...,d_u \in \{0,1,2\},\ \text{and } 0 < \varepsilon_1 < 3^{-B_1}5^{-u};$$

Moreover, if we write

$$\gamma\left(\frac{1}{3^{x_1}}+\frac{1}{3^{x_2}}+\cdots+\frac{1}{3^{x_t}}\right)+\delta$$

in its base-7 expansion,

$$\frac{e_1}{7} + \frac{e_2}{7^2} + \cdots + \frac{e_v}{7^v} + \varepsilon_2,\ v = \lceil x_t \log(3)/\log 7 \rceil, \tag{2}$$

then

$$e_1, e_2, ..., e_v \ \in\ \{0, 1, 2, 3\},\ \text{and}\ 0\ <\ \varepsilon_2\ <\ 3^{-B_1}7^{-v}.$$

This appears to be an infinite problem, since $\alpha, \beta, \gamma, \delta$ are allowed to be real numbers; however, it only suffices to verify the above for all such variables looking at the top few base-3 digits – that is, we may assume that $\alpha, \beta, \gamma, \delta$ have the form

$$\frac{f_1}{3} + \frac{f_2}{3^2} + \cdots + \frac{f_{B_2+m}}{3^{B_2+m}}.$$

From here, we show that if the working hypothesis in the previous theorem statement holds for a pair of integers $B_1 < B_2$, then we must have that it holds for the same B_1 and where we can choose B_2 as large as we wish. To establish this, we will show that we may simple enlarge the value of B_2 – and thus, we can iteratively enlarge it as much as desired. This argument provides the foundation of our exhaustive algorithmic computer search which gives a proof of Erdös's long standing conjecture in a very modern way. That is, provided we can find an effective upper bound on the initital integer B_1 which we can extend which yields computationally feasible super computer searches to enumerate all possible cases we need to consider here. We treat this upper bound using a heuristic which we can then refine within our algorithm in the sections below.

2 A proof of the initial working hypothesis

We provide a proof of the working hypothesis stated in Theorem 1 by setting up a system of non-linear systems for the integers $X^{[t]}(\alpha, \gamma, \beta, \delta) := \{x_i\}_{i=1}^{t}$ and reducing with the constraint that all of our expansions are considered to be truncated base-5 or base-7 expansions of reals in $(0, 1)$ with digits reduced modulo each p. The nature of the non-linear equations introduced by our initial expansions result in polynomials formed by the binomial coefficients each of which have simple, or at least predictable, rational roots as polynomials of each x_i. We then establish a system of these equations with constrained parameters for each fixed tuple of $(\alpha, \gamma, \beta, \delta)$. We show the existence of an integer $t \geq 1$ and the $1 \leq x_1 < x_2 < \ldots < x_t < B_1$.

2.1 Existence of solutions to constrained non-linear systems modulo p

Lemma 2. ***TODO: Hamed / Hensel's lemma / integer programming / etc.***

Proof. □

2.2 Assembling the proof of our working hypothesis

Proof of Theorem 1. Fix a 4-tuple $(\alpha, \gamma, \beta, \delta)$ subject to the constraints in the theorem statement. Let

$$\alpha := \sum_{j \geq -u_\alpha} a_j(\alpha)3^{-i}, \beta := \sum_{j \geq 1} b_{5,j}(\beta)5^{-j}, \delta := \sum_{j \geq 1} d_{7,j}(\delta)7^{-j},$$

where $\widetilde{a}_i(\alpha) \in \{0,1,2\}$ for all i and where $b_{5,j}(\beta) \in \{0,1,2,3,4\}$ and $d_{7,j}(\delta) \in \{0,1,2,3,4,5,6\}$ for all $j \geq 0$. Notice that these expansions may result in infinite base-p expansions for $p = 3, 5$. Now we may proceed inductively to construct a complete satisfactory set of $X^{[t]}(\alpha, \gamma, \beta, \delta)$ from initial constraints on some singleton set $\{x_1\}$. We must first show that for some $t \geq 1$, there are t distinct (increasing) integers x_i such that the claim in (1) is satisfied.

Transforming parameters and formulating constraints. Observe next that we can compute that

$$\begin{aligned}
f_1^{[t]}(\alpha, \beta) &:= \alpha \left(\frac{1}{3^{x_1}} + \cdots + \frac{1}{3^{x_t}} \right) + \beta \\
&= \alpha \times \sum_{i=1}^{t} \sum_{n \geq 0} \binom{n + x_i - 1}{x_i - 1} \frac{2^n}{5^{n+x_i}} + \beta \\
&= \sum_{j \geq -u_\alpha} \sum_{i=1}^{t} \sum_{n \geq 0} a_j(\alpha) \binom{n + j + x_i - 1}{x_i - 1} \frac{2^n}{5^{n+j+x_i}} + \beta. \\
f_2^{[t]}(\gamma, \delta) &:= \gamma \left(\frac{1}{3^{x_1}} + \cdots + \frac{1}{3^{x_t}} \right) + \delta \\
&= \gamma \times \sum_{i=1}^{t} \sum_{n \geq 0} \binom{n + x_i - 1}{x_i - 1} \frac{4^n}{7^{n+x_i}} + \delta \\
&= \sum_{j \geq -u_\gamma} \sum_{i=1}^{t} \sum_{n \geq 0} a_j(\gamma) \binom{n + j + x_i - 1}{x_i - 1} \frac{4^n}{7^{n+j+x_i}} + \delta
\end{aligned}$$

Now set

$$\binom{n + x - 1}{x - 1} 2^n := \sum_{m \geq 0} f_{x,n}^{(1)}(m) 5^m,$$

and

$$\binom{n + x - 1}{x - 1} 4^n := \sum_{m \geq 0} f_{x,n}^{(2)}(m) 7^m,$$

where these base-$5, 7$ expansions are finite. Then we can write the previous equations in the following form:

$$\begin{aligned}
f_1^{[t]}(\alpha, \beta) &= \sum_{j \geq -u_\alpha} \sum_{i=1}^{t} \sum_{n \geq 0} \sum_{m \geq 0} \frac{a_j(\alpha) \cdot f_{x_i, n+j}^{(1)}(m)}{5^{n+j-m+x_i}} + \sum_{j \geq 1} b_{5,j}(\beta) 5^{-j} \\
f_2^{[t]}(\gamma, \delta) &= \sum_{j \geq -u_\gamma} \sum_{i=1}^{t} \sum_{n \geq 0} \sum_{m \geq 0} \frac{a_j(\gamma) \cdot f_{x_i, n+j}^{(2)}(m)}{7^{n+j-m+x_i}} + \sum_{j \geq 1} d_{7,j}(\beta) 7^{-j}
\end{aligned}$$

Let the base-5 expansion indexing sets, I_N, be defined by

$$I_{N,i} := \{(i, n, j, m) \in \mathbb{Z}^+ \times \mathbb{N} \times \mathbb{Z} \cap [-u_\alpha, \infty) \times \mathbb{N} : n - j - m + x_i = N\},$$

and let the carry function be defined by

$$C_p^{(k)}(\alpha_0, \beta_0; N, i) := \sum_{(x,i,n,j,m)\in I_{N,i}} a_j(\alpha_0)\cdot f_{x,n+j}^{(1)}(m) + b_{p,N}(\beta_0) - \sum_{s=1}^{k} \left\{ \sum_{(x,i,n,j,m)\in I_N} a_j(\alpha_0)\cdot f_{x,n+j}^{(1)}(m) + b_{p,N}(\beta_0) \pmod{p^s} \right\}.$$

So what we have in effect is the system of equations and constraints which prescribe that for all $0 \leq k < N \leq \max(u, v) = u$ and $1 \leq i \leq t$, each of the following additional conditions must be satisfied:

$$\sum_{(i,n,j,m)\in I_{N,i}} a_j(\alpha) f_{x,n+j}^{(1)}(m) + b_{5,N}(\beta) \bmod 5 \in \{0,1,2\} \quad (3)$$

$$\sum_{i=1}^{t} \left[\sum_{(i,n,j,m)\in I_{N,i}} a_j(\alpha) f_{x,n+j}^{(1)}(m) + b_{5,N-k}(\beta) + \sum_{j=1}^{20k} C_5^{(j)}(\alpha, \beta; N-k+j, i) \right] \bmod 5 \in \{0,1,2\}$$

$$\sum_{(i,n,j,m)\in I_{N,i}} a_j(\gamma) f_{x,n+j}^{(2)}(m) + d_{7,N}(\delta) \bmod 7 \in \{0,1,2,3\}$$

$$\sum_{i=1}^{t} \left[\sum_{(i,n,j,m)\in I_{N,i}} a_j(\gamma) f_{x,n+j}^{(2)}(m) + b_{7,N-k}(\delta) + \sum_{j=1}^{36k} C_7^{(j)}(\gamma, \delta; N-k+j, i) \right] \bmod 7 \in \{0,1,2,3\}.$$

For each fixed t, this leads to a system of

$$\#(t) = \frac{1}{2}\left\lceil x_t \frac{\log(3)}{\log(5)} \right\rceil \left(\left\lceil x_t \frac{\log(3)}{\log(5)} \right\rceil + 1 \right),$$

constraints which must be satisfied by some concrete prescribed sequence of increasing integers $X^{[t]}(\alpha, \gamma, \beta, \delta)$. We notice a small subtlety in the proof of the existence of this list of integers which is that as we are trying to optimize and fix constraints on the x_i's by our computerized search, the number of constraint equations are implicitly varying with the increasing sequence of x_t. It is not possible in this case to simply assume a fixed indeterminate of the x_t since broadening the search space on the other x_i for $1 \leq i < t$ necessarily ramps up the lower end of values which may be assumed by x_t.

Existence of solutions. Now we show that for any fixed 4-tuple selected as above, we must have some integer $t \geq 1$ and corresponding set $X^{[t]}(\alpha, \gamma, \beta, \delta)$ which satisfies the constraints set forth by the above expansions. By Lemma 2, we see that ... (**TODO**)

Showing that an initial finite B_1 can be selected is non-trivial. Finally. note that an argument for the existence of such a bounded integer B_1 such that the theorem statement holds is shown to exist in the argument given in Section 3.3 below. More precisely, we show that given a small pickup in savings on the number of searches of the 4-tuples which we have to perform each time we compute new data for any given 4-tuple we process, i.e., if we have a sane and well-planned programatic structure via dynamic programming behind our algorithm, then the upper bound on B_1 can be shown to converge to a finite limiting case as the number of tuples considered is large. We cannot explore this last justification step in the proof at this point. Instead, we delay our calculations of this convergence fact until we have reasoned more about the problem and it's inherent computational aspects in the next sections. □

3 A heuristic upper bound on the initial integer B_1

3.1 Overview to the method

In this section, we are going to explore an adaptive approach to refining and/or tractibly bounding the value of B_1 in that we need to find an effective working upper bound on our search space by determining a concrete integer B_1, but it must also be tight enough to the actual minimal lower bound that would work here so that we can in fact run our computations on a modern Linux PC in less than a week. We will specify the constraints to the problem in the search space over the truncated real-valued $(\alpha.\gamma, \beta, \delta)$ from Theorem 1 is in fact finite. In the interim, we will specify a priori that such a (large) B_1 must exist and be some bounded finite integer with respect to this problem, but allow the dimensions of our search space to be implicitly defined by B_1 and the other parameters, such as $t_{\max}$, from the working hypothesis. We will then construct a provable method to lower the implicit parameter bounds at each iteration processing a 4-tuple in our search space until the effective size of the search space is either 1) below our integer maximum threshold, $I_{\max} = 2^b - 1$; or 2) Until we can guarantee by convergence of our reduction method that we have reached an optimally minimal value of B_1 that can be reduced no further.

For our purposes, we can take a tractable value of the $I_{\max} \sim 2^{45}$ arithmetic or floating point operations[1]. We will also analyze the affect of sorting, or randomizing the iterative selection of the 4-tuples, $(\alpha.\gamma, \beta, \delta)$, from the remaining possibilities in our finite search space as we process each combination of the values. As a part of the verification method, we must ensure that our algorithm produces a, i.e., at least one, *certificate* list of $\{x_1, x_2, \ldots, x_t\}$ $(t \geq 1)$ for each tuple of the real numbers in our search space. Then if we can prove by convergence or bounds on our algorithmic process that the search space of these real paramaters is finite, these certifcates will constitute the computerized "proof" of the Erdös conjecture for $\gcd(\binom{2n}{n}, 105)$ by extension to the further $B_2 > B_1$ ad infinitem (see the next section).

3.2 A guiding heuristic for implicit upper bounds on B_1

We continue along citing the notation specified in the statement of the working hypothesis from Theorem 1. Let N denote the current size (cardinality) of the adaptive search space after some number of iterations through our algorithm. We can easily overestimate the maximum integer parameter $t_{\max}$ such that $1 \leq x_1 < \cdots < x_t < B_1$ by $1 \leq t \leq t_{\max}$ for $t_{\max} = B_1$. Then we can determine that there are at most the following number of relevant expansions of the sum $\alpha(3^{-x_1} + \cdots + 3^{-x_t}) + \beta$:

$$N = t_{\max} \cdot B_1 \cdot \lceil \log_3(u_\alpha) \rceil + B_1 \implies B_1^2 \lceil \log_3(u_\alpha) \rceil + B_1 - N \geq 0.$$

[1]We will assume a processor speed clocked at 3 Ghz, or 3×10^9 cycles per second, where as is reasonable to expect from [2] that each arithmetic or floating point operation we perform takes an average of say 70 cycles to perform at average latency. Then we require that

$$\frac{1 \text{ second}}{3 \times 10^9 \text{ cycles}} \times 2^N \text{ operations} \times \frac{70 \text{ cycles}}{1 \text{ operation}} \leq \frac{7 \cdot 24 \cdot 3600 \text{ seconds}}{1 \text{ week}} \implies N \leq \log_2(2.592 \times 10^{13}) \approx 44.6 \text{ operations}.$$

A more exhaustive analysis would of course depend on the actual hardware implementations and the sequence of the assembly language instructions we choose in our implementation of the algorithm. This estimate provides us with a reasonable upper bound to shoot for in constructing our algorithm below.

Input : Bounds on the effective search space for the real numbers $1 \leq \alpha \leq u_\alpha$ and $1 \leq \gamma \leq u_\gamma$. Let's call it $\mathbb{R}_{105,\text{trunc}} \subseteq \mathbb{R}^4_{>0}$, which has a nice bounded finite cardinality of some integer $N \sim 3^{20}$.

Output: A certificate list of plausible integers $\{x_1, x_2, \ldots, x_t\}$ for each tuple $(\alpha, \gamma, \beta, \delta) \in [1, u_\alpha] \times [1, u_\gamma] \times (0, 3^{-B_1})^2$ such that our working hypothesis holds. Here, the search space for the truncated base-p expansions of these real 4-tuples must be finite and predictably bounded.

Figure 1: Constraints and initial assumptions on our search space. *We specify these standard definitions and fixed parameters specifying the ranges on our search space of real-values 4-tuples which we must evaluate to prove the conjecture in practice.*

Since $B_1 \geq 1$ and $N \geq 1$, we see that this implies that

$$B_1 \approx \left\lceil \frac{\sqrt{4N\lceil \log_3(u_\alpha)\rceil + 1} - 1}{2\lceil \log_3(u_\alpha)\rceil} \right\rceil,$$

which in turn implies that we have at most

$$3^u = 3^{B_1 \left\lceil \frac{\log(3)}{\log(5)} \right\rceil} \approx 3^{\left\lceil \sqrt{\frac{N}{\lceil \log_3(u_\alpha)\rceil}} \right\rceil \cdot \left\lceil \frac{\log(3)}{\log(5)} \right\rceil},$$

total valid cases of the truncated base-5 expansions of $\alpha(3^{-x_1} + \cdots + 3^{-x_t}) + \beta$ which we need to consider in our search. Similarly, we can obtain that there are at most

$$4^v = 4^{B_1 \left\lceil \frac{\log(3)}{\log(7)} \right\rceil} \approx 4^{\left\lceil \sqrt{\frac{N}{\lceil \log_3(u_\gamma)\rceil}} \right\rceil \cdot \left\lceil \frac{\log(3)}{\log(7)} \right\rceil},$$

total possible cases of the truncated expansions of $\gamma(3^{-x_1} + \cdots + 3^{-x_t}) + \delta$ we need to consider in our search.

3.3 An exact procedure for finding a tractable $B_1 \leq I_{\max}$ by adaptive search

Proposition 3 (Parameter Bounds and Convergence of an Adaptive Search for B_1)**.** *Suppose that we can reduce the size of the search space from N to $(1-r) \cdot N$ for some $r \in (0,1)$ using previously computed, dynamically stored data at each iteration over the 4-tuples in the algorithm. Then provided that $u_\alpha \geq 43$ and $u_\gamma \geq 60$, we get the convergence of B_1 to a limiting finite value for any fixed initial search space upper bound N_0.*

Proof. Under the assumption of an expected size of the search space after an iteration over a fixed 4-tuple, or let's say more efficiently after processing a sufficiently large batch of tuples in the search space as a single "round" of the search procedure, we can apply the ratio test to find sufficient conditions on the adaptive convergence of B_1 to a limiting tight upper bound. In particular, since $\sqrt{1-r} - 1 < 0$ by the ratio test and the approximations in the last subsection, we can write an approximate expression for $B_1(N)$ as

$$B_1(N) \leq C_1^{\sqrt{N}} + C_2^{\sqrt{N}},$$

where

$$C_1 := 3^{\frac{\log(3)}{\log(5)\sqrt{\log_3(u_\alpha)}}}, C_2 := 4^{\frac{\log(3)}{\log(7)\sqrt{\log_3(u_\gamma)}}}.$$

Then after one iteration of processing some 4-tuples so that we reduce the search space from $N \mapsto (1-r)N$, we can compute the ratio of consecutive $B_1(N)$'s is given by

$$\frac{1}{1+\left(\frac{C_1}{C_2}\right)^{\sqrt{N}}} \cdot \left(\left(\frac{C_1^{1-r}}{C_2}\right)^{\sqrt{N}} + C_2^{(\sqrt{1-r}-1)\sqrt{N}}\right).$$

If we then select the so far undetermined parameters u_α, u_γ so that $C_1, C_2 > \frac{3}{2} > 1$ and $\frac{C_1}{C_2} < \frac{3}{2} < 1$, we have that the ratio converges to zero as $N \to \infty$. We can satisfy the inequalities above by computing numerically that these requirements hold for $u_\alpha := 43$ and $u_\gamma := 60$. $\square$

3.3.1 Obtaining effective upper bounds on B_1 and a perspective on the computational order of this problem

Proposition 4 (Obtaining the Stated Lower-Upper Bound on B_1)**.** *If we select B_1 in the working hypothesis from Theorem 1 as an initial value which satisfies the hypotheses above, it suffices to consider maximally that $B_1 := 5$.*

Proof. If we let as before, N denote the finite size of our search space over the real 4-tuples, we can count that

$$N \leq 2t_{\max}(x_{t_{\max}} + \max(\log_3(u_\alpha), \log_3(u_\gamma))) \leq 2B_1(B_1 + \log_3(60)).$$

Additionally, by overcounting the product of the possibilities for digits in (1) and (2), we can similarly estimate that this right-hand-side bound is also less than or equal to

$$3^{B_1\frac{\log(3)}{\log(5)}} \times 4^{B_1\frac{\log(3)}{\log(4)}}.$$

Now by taking logarithms, simplifying, and then numerically computing bounds on the B_1 which satisfy these inequalities, we are able to establish that $-3.74205 \leq B_1 \leq 0.314955 \vee B_1 \geq 4.8688$. So by the constraints defined by our problem it suffices to consider B_1 only as large as five here. $\square$

Corollary 5 (Time Complexity of Our Algorithm)**.** *We can perform the exhaustive brute force search in our algorithm which enumerates all possible cases of the 4-tuples we must consider on the order of at most 3^{28} arithmetic or floating point operations.*

Proof. First, we notice that by Proposition 4 we have a nicely constrained small set of cases of the possibilities for the integers $\{x_1, \ldots, x_t\}$ that we must check for each distinct fixed 4-tuple of reals in our search space. Namely, there are only $2^5 - 1 = 31$ possibilities for these resulting integer sets. Since this does not add a prohibitively expensive constant multiplier on the running time of our application if we simply check them all for a successful match at each fixed 4-tuple. This conveniently moves our strategy to the computational task in this problem to a brute force methodology in place of having to carefully leverage a more existence-like consideration of such cases to reduce the running time. This is one factor to the claimed order of the time complexity in our algorithm.

Next, we must compute effective finite upper bounds on the number of possibilities for each tuple parameter $\alpha, \gamma, \beta, \delta$ we must factor into the running time. We use the given bounds in our working hypothesis in the forms provided by Proposition 3. The β, δ cases are the most straightforward to produce bounds for, so we will determine how they factor into the running time before the other two cases. Notice that by (1) we need only consider distinct cases of the non-negligible base-5 expansions of the β after truncating the expansion at the powers of u from Theorem 1. Since we have set $B_1 := 5$ according to the previous proposition, this implies that we need only enumerate all of the acceptable base-5 digits of the powers of 5^{-j} for $1 \leq j \leq \left\lceil 5\frac{\log(3)}{\log(5)} \right\rceil = 4$. Thus here we have a total of $3^4 = 81$ cases of the digit expansions of β to consider in our search algorithm. Similarly, we obtain that there are at most $4^{\left\lceil 5\frac{\log(3)}{\log(7)} \right\rceil} = 64$ distinct truncated digit expansions of the δ which we need to enumerate.

For the case of enumerating the distinct forms of the α in our search, we note that we can selectively restrict ourselves to processing the cases of $\frac{\alpha}{3^{x_i}} < 1 \implies \left\lceil \frac{\log_3(\alpha)}{\log(3)} \right\rceil < x_i$. Otherwise if this bound does not hold for all of the x_i when α is fixed, then we can safely move along in the algorithm without further processing. Now since an upper bound on α is shown to be $u_\alpha := 43$ in Proposition 3, by excluding the sets of $\{x_i\}$ which do not satisfy the previous property, we can conclude that we have a maximum of $3^{\lceil \log_3(43) \rceil + 1} = 3^5$ distinct digit configurations for the integer parts of the α we need consider here. To determine the corresponding digit configuration count for the fractional parts of these α, we consider the expansions

$$\alpha\left(3^{-x_1} + \cdots + 3^{-x_t}\right) = \sum_{j \geq -u_\alpha} \sum_{i=1}^{t} \frac{a_j}{3^{x_i+j}} = \sum_{j \geq -u_\alpha} \sum_{i=1}^{t} \sum_{n \geq 0} \binom{x_i + j + n - 1}{n - 1} \frac{2^n a_j}{5^{x_i+j+n}}.$$

So we can overestimate the range of postive j by ignoring the contributions of the positive sum of $x_i + n$ and establishing that $1 \leq j \leq u \leq \left\lceil \frac{5\log(3)}{\log(5)} \right\rceil = 4$. We conclude that there are actually only 3^4 cases of the fractional parts of α. In total, this leads to the finite bound of 3^9 distinct truncated digit expansions of α to enumerate in our search. A nearly identical argument for the γ shows that there are $4^{5+3} = 4^8$ distinct cases to consider.

Then in total, our upper bound on the running time in number of operations required corresponds to the product of these bounds: $31 \cdot 81 \cdot 64 \cdot 3^9 \cdot 4^8 = 2.073 \times 10^{14} \sim 3^{31} \approx 2^{48}$. $\square$

4 Stitching together loose ends: finalizing a constructive proof of the correctness of our algorithm

4.1 Expanding to a larger value of $B_2 > B_1$ iteratively

4.1.1 The iterative procedure by extension

Suppose the claim of Theorem 1 proved in the previous section holds for the pair $B_1 < B_2$, let $\alpha, \beta, \gamma, \delta$ be any constants satisfying the constraints of that hypothesis; and let $x_1, ..., x_t$, u, v, and ε_1 and ε_2 be as indicated. Now let

$$\alpha' := \alpha 3^{-x_t} 5^u \in [1, u_\alpha), \ \beta' := \varepsilon_1 5^u \in [0, 3^{-B_1}).$$

and let

$$\gamma' := \alpha 3^{-x_t} 7^v \in [1, u_\gamma), \ \delta' := \varepsilon_2 7^v \in [0, 3^{-B_1}).$$

Then we apply the hypothesis to these new choices for $\alpha, \beta, \gamma, \delta$, and find that there exist $1 \leq y_1 < y_2 < \cdots < y_z \leq B_2$ as the new sequence of x_i's.

4.1.2 Justification for why this works

We can then combine our two lists of numbers together as follows: we leave $x_1, ..., x_t$ as it was before; but now set

$$x_{t+i} \ := \ x_t + y_i.$$

So, we have that

$$\alpha\left(\frac{1}{3^{x_1}} + \frac{1}{3^{x_2}} + \cdots + \frac{1}{3^{x_{t+z}}}\right) + \beta$$

can be written as

$$\alpha\left(\frac{1}{3^{x_1}} + \cdots + \frac{1}{3^{x_t}}\right) + \beta + \alpha' 5^{-u}\left(\frac{1}{3^{y_1}} + \cdots + \frac{1}{3^{y_z}}\right).$$

This expression may, in turn, be written as

$$\frac{d_1}{5} + \cdots + \frac{d_u}{5^u} + \frac{d_{u+1}}{5^{u+1}} + \cdots + \frac{d_{u'}}{5^{u'}} + \varepsilon_1 - 5^{-u}\beta' + 5^{-u}\varepsilon_1',$$

where $\varepsilon_1' < 3^{-B_1}5^{-u'+u}$. Since $\varepsilon_1 = 5^{-u}\beta'$, this simplifies to

$$\frac{d_1}{5} + \cdots + \frac{d_{u'}}{5^{u'}} + \varepsilon_1'',$$

where $\varepsilon'' < 3^{-B_1}5^{-u'}$. And, again, we have that all the $d_i \in \{0,1,2\}$. The analogous result hold for the base-7 expansion. So, we conclude that if we find a viable pair $B_1 < B_2$, we can expand B_2 indefinitely.

4.2 The working hypothesis implies that $\mathcal{B}_{3,5,7}$ is infinite

We will use the claim from the previous section to algorithmically construct integers n with the desired property. That is, if we have shown that there is a viable starting pair of integers $B_1 < B_2$ satisfying the working hypothesis, then we can inductively construct another $B_2 < B_3$ and so on with the properties in Theorem 1. Rather than list out the exact steps of this algorithm here, we will here only present how the first two steps work – the other steps just repeat the second step (see Section 5). We hinge our argument on the well-known fact stated explicitly in the next theorem due to Kummer:

Theorem 6 (Kummer). *We have that* $\gcd(\binom{2n}{n}, 105) = 1$ *for n a positive natural number if and only if the base-3 digits of n are in $\{0,1\}$, the base-5 digits are in $\{0,1,2\}$, and the base-7 digits are in $\{0,1,2,3\}$. More generally, if $p_1, p_2, \ldots, p_n$ are odd primes, then* $\gcd(\binom{2n}{n}, p_1p_2\cdots p_n) = 1$ *for $n \geq 1$ if and only if the digits in the base-p_i expansion of n are all in the range $\left\{0, 1, 2, \ldots, \left\lfloor \frac{p_i}{2} \right\rfloor\right\}$ for all $1 \leq i \leq n$.*

4.3 A sketch of our complete algorithmic procedure

- **Step 1.** Let $B_1 < B_2$ be a pair of integers satifying the initial working hypothesis. We begin by selecting $h \geq 1$ to be any integer such that

$$|3^h - 5^k| \ < \ 3^{-B_1}, \text{ and } |3^h - 7^\ell| \ < \ 3^{-B_1},$$

where B_1 is as in the previous section and where

$$k = \lfloor h \log(3)/\log 5 \rfloor, \text{ and } \ell = \lfloor h \log(3)/\log 7 \rfloor.$$

This is just describing that the upper few digits of 3^h in base 5 and base 7 begin with 1, followed by a few 0's.

- **Step 2.** Next, we set
$$\alpha := 3^h 5^{-k}, \text{ and } \gamma := 3^h 7^{-\ell}.$$
and set
$$\beta := \delta := 0,$$
Applying the "*working hypothesis*" of Theorem 1 in the form of it's implications from the previous section to $\alpha, \beta, \gamma, \delta$ we find that there exist $B_1 < B_2$, with B_2 arbitrarily large, and $x_1, x_2, ..., x_t$ satisfying the conclusion of that claim. This implies that
$$3^{h-x_1} + \cdots + 3^{h-x_t} = 5^k(\alpha(3^{-x_1} + \cdots + 3^{-x_t}) + \beta) = d_1 5^{k-1} + d_2 5^{k-2} + \cdots + d_u 5^{k-u} + 5^k \varepsilon_1,$$
where $d_1, d_2, ..., d_u \in \{0, 1, 2\}$, and $5^k\varepsilon_1 < 3^{-B_1} 5^{k-u}$ – so, the lower digits in base-5 our of sum of powers of 3 (below the coefficients of 5^{k-u}) will be 0 for a while, then it could be something different. We must perform an exhaustive search of all cases to find out.

- **Step 3.** Similarly, we have
$$3^{h-x_1} + \cdots + 3^{h-x_t} = e_1 7^{\ell-1} + \cdots + e_v 7^{k-v} + 7^\ell \varepsilon_2,$$
where $e_1, ..., e_v \in \{0, 1, 2, 3\}$, and $7^\ell \varepsilon_2 < 3^{-B_1} 7^{\ell-v}$.

- **Conclusion.** So, the upper few digits of $3^{h-x_1} + \cdots + 3^{h-x_t}$ look good bases $3, 5$, and 7. If B_2 is large enough, this will cover all the base-5 and 7 digits below 5^0 and 7^0, which gives us an n whose base-3, base-5 and base-7 digits are such that there are no carries when adding the number to itself in these bases.

5 Results of the computational step in our proof

We have written a mostly efficient, though not too finely optimized, implementation of our algorithm which we have rigorously justified to be correct in C++. We have limited the scope of our optimizations with the source code for readbility purposes, though we point out that further low-level optimizations and fine-tuning through, for example, assembly languages or libraries which make use of the readily available floating point processing power found in most moder GPUs are possible. In fact, such low-level programatic tweaking of the open source implementation we provide in [4] may allow further extensions of our algorithmic process to be generalized and immediately *effectively proved* for the gcd of $\binom{2n}{n}$ and products of other odd prime triples and k-tuples. We begin with the simplified pseudocode sketch of our working algorithm in practice given in the next listing (see Figure 2).

Using this algorithm, we have performed the explicit computations involved in verifying Erdös's conjecture in this case in roughly 3^{21} operations. The computations in question were performed using the super computer resources in the mathematics department at the Georgia Institute of Technology in October / November of 2018. The runtime of our program executed in approximately

```
Input  : A computationaly tractable upper bound on the initial $B_1$.
Input  : Bounds on the effective search space for the real numbers $1 \leq \alpha \leq u_\alpha$ and
         $1 \leq \gamma \leq u_\gamma$. Let's call it $\mathbb{R}_{105,\mathrm{trunc}}$, which has a nice bounded finite cardinality of
         some integer $N \sim 3^{21}$.
Output: A certificate list of plausible integers $\{x_1, x_2, \ldots, x_t\}$ for each tuple
         $(\alpha, \gamma, \beta, \delta) \in [1, u_\alpha] \times [1, u_\gamma] \times (0, 3^{-B_1})^2$ such that our working hypothesis holds.
         Here, the search space for the truncated base-$p$ expansions of these real 4-tuples
         must be finite and predictably bounded. Upon successful completion of the
         algorithm without interruption, the return value is a boolean-valued True
         affirmative of the infinitude of $\mathcal{B}_{3,5,7}$. Otherwise, a value of False is returned to
         indicate a failed search.
def RealSearchSpace ⟵ ThoughtfulSortIterator($\mathbb{R}_{105,\mathrm{trunc}}$);
def HashTable ⟵ InitializeNewLookupTable();
def CurrentB$_1$ ⟵ InitialUpperBoundB$_1$;
def LastUpdatedB$_1$ ⟵ CurrentB$_1$;
def $N$ ⟵ CurrentB$_1^2$ $\lceil \log_3(u_\alpha) \rceil$ + CurrentB$_1$;
for $1 \leq n \leq N$ do
    agbdTuple ⟵ RealSearchSpace.next();
    $t, \{x_1, \ldots, x_t\}$ ⟵ CertificateSearch(abgdTuple, RealSearchSpace, HashTable);
    call PrintCertificate(abgdTuple, $t, \{x_1, x_2, \ldots, x_t\}$);
    $N$, CurrentB$_1$ ⟵ RealSearchSpace.getNextSizeParameters();
    if CurrentB$_1$ − LastUpdatedB$_1$ ≥ UpdateDifferenceBoundB$_1$ then
        PruneDynamicStorageSpace(HashTable);
    end
    if CurrentB$_1$ ≤ $I_{\max}$ then
        break;
    end
    else if RealSearchSpace.isEmpty() then
        return False;
    end
end
return CompleteBruteForceSearchInRange(CurrentB$_1$, RealSearchSpace, HashTable);
```

Figure 2: Simplified pseudocode for our algorithm
The listings in the previous numbered lines provide an approximately C-like algorithmic syntax for the description of our working procedure. Where possible, we have simplified or abstracted the lower-level details to the actual C++ implementation we ran in the fall of 2018 at Georgia Tech. The total line count as of the submission of this manuscript for the complete compilable source code to our iterative search-verify-and-extend routines is 1000 (TODO) lines of ascii-formatted plaintext (see [4]).

3.2 (TODO) days running on a quad-core Ubuntu Linux desktop (TODO) machine with 3.4 Ghz (TODO) processors and TODO GB of RAM space. Our resulting conclusive computations, did in fact, return the expected affirmative response on the infinitude of the set $\mathcal{B}_{3,5,7}$ which we defined in the introduction. It is thus our pleasure to conclude our modern day proof with a classical *QED* for the problem. □

6 Conclusions and generalizations of our method

6.1 A summary and some concluding remarks

The first author of this manuscript has expressed reservations as to whether we will be able to claim the cash prize offered by Ron Graham for the solution to this problem via a proof by exhaustive computerized search. The remaining authors have a more optimistic view on the role of computer algorithms and algebra packages in assisting experimental mathematics in the general case. Well-known algorithms for exactly summing hypergeometric series, such as those predicted by the WZ algorithm and others, have been known to researchers in the field for many years. These translations of important mathematical problems into a finite-time algorithm process which is readily accessible through affordable modern computer systems has arguably been a transformative process in our discipline of fundamental importance over the last few decades. In the same spirit, we hope that our translation of Erdös's problem on fundamental congruence properties of the binomial coefficients into a (non-linear) integer programming task will inspire others to prove a much richer, more general class of gcd properties and congruence relations for subsequences of the binomial coefficients in the future.

Our computerized method of automating the enumeration process of an initial base set of integers which can be searched, and checked for property X exhaustively in predictably finite time is new progress in the way of examining such previously unattainable congruences for this classical combinatorial triangle. Put another way, our success in proving Erdös's famous 105-conjecture here is to rigorously show that the infinitude of the natural numbers with some property X (here our key requirement that $\gcd(\binom{2n}{n}, 105) = 1$) holds if some humanly intractable, but finite set of cases can be conclusively checked by modern brute-force computational rigor. We have reduced our logic to finding the initial bounded B_1 and then iteractively extending, and in the process have relegated the chore of producing binary and base-p expansions of integers and selected real numbers to the tool that understands them best. In this way, we view the computational extension of our thought process on this problem to be a significant new step in the right direction towards classifying a larger subset of classically posed and studied binomial coefficient congruence properties.

Surely this extendable method will provide other experienced number theorists and computational researchers with a new set of approaches to famous long standing open problems – if not capture the cash award which when split three ways offers the authors each a new portable computer gadget. *Procedo at ad (in)finitem!*

6.2 Unstated applications and future work on the relative primality of the binomial coefficients by products of odd primes

We remark that the generalized form of Theorem 6 for $n \geq 3$ offers an immediate generalization of our computerized search method by simply changing the constants in our formulas here. The only caveat of this observation when $n \geq 4$ which complicates finding a rigorous proof of the correctness of the proposed algorithmic search method is that adding further constraints to the list in (3) necessarily makes the existence result we proved in Lemma 2. Making this mathematically involved jump, perhaps by induction, provides a new challenge to readers and reviewers of this article. In particular, proving analogous forms of the working hypothesis which can be shown to hold modulo arbitrary products of odd primes is a significant open problem which we believe can now be approached by a hybrid of computation and rigorously justified formal mathematics.

6.3 A discussion of optimizations and adaptations to the algorithm

At this point we will highlight some of the less mathematical components to making our algorithm computationally feasible in more general cases. We point out that some of our time and gradual work on this proof involved planning for making a computationally intensive task in processing all possible cases in the base set for B_1 more feasible. It turns out that our specific problem is tractable enough to be run in a reasonable amount of time given a new Linux PC. In the event that attempts to generalize our results and method of proof modulo products of a larger number of odd primes lead to computationally infeasible calculations, we share some of our thoughts and preparation here. The remarks that follow are by no means comprehensive and suffice to provide a rough treatment of these topics which may be perfected in later generalizations by more careful consultations with experts in computer science and engineering disciplines. In particular, we note that more formal treatments to each of these topics can be found in print, for example, in [1, 6].

First of all, an implicit reliance on the mechanism of dynamic programming and special purpose hash table data structures is a must in the general case. These techniques are aimed at strategically reducing the amount of recomputation of calculations by caching the data obtained in previous iterations of the algorithm so that it can be reused. This requires some thought and insights into the best ways to process tuples in the search space since RAM and CPU cache memory are limited. Our initial plans for writing a tractable algorithm for the 105-problem touched on how the order in which we iterate over all possible adaptations of these real 4-tuples can affect the running time of our implementation in practice. For example, a non-obvious technical question along these lines is to determine a “best” or optimal search strategy which in expectation on the average case leads to an acceptably low number of cache misses and recomputations we have already done before.

One way to handle this problem is to consider the effects of randomization on the order of selection of the tuples in our search space. This approach certainly suggests that the resulting code could be more readily parallelized to process say all tuples within an interval selected uniformly at random of size say some M such that this procedure is designed to give a more efficient overall running time when processing the entire search space. An effective algorithm for the general case will most likely require an adaptation of the formal mathematics to our existence proof of the parameters in the working hypothesis, and new insights into the underlying problem structure which optimize the running times and space complexity required by the computerized search-and-conquer algorithm we have devised in this article.

Acknowlegements

Hamed Mousavi and Maxie D. Schmidt thank our adviser, the first author, for generously posing a digestable formulation of the algorithm in question to us as a collaborative research project. Professor Croot has had working knowlege of this solution, minus wielding the modern computational resources which we have completed here, for more than a decade. With the advent of large-scale implementations of multicore modern PC machines, we have finally arrived at a satisfying affirmative conclusion to Erdös’s problem.

References

[1] M. P. Bekakos, *Highly parallel computations: algorithms and applications*, WIT Press (2001).

[2] A. Fog, Instruction tables: Lists of instruction latencies, throughputs and micro-operation breakdowns for Intel, AMD and VIA CPUs, `https://www.agner.org/optimize/instruction_tables.pdf` (2018).

[3] C. Pomerance, Divisors of the middle binomial coeficient, Amer. Math. Monthly (2014).

[4] M. D. Schmidt, GitHub source code repository for the implementation of our 105 algorithm, available online at `https://github.com/maxieds/Erdos105Problem`.

[5] N. J. A. Sloane, Sequence A030979 in the Online Encyclopedia of Integer Sequences, `https://oeis.org/A030979` (2018).

[6] A. S. Tanenbaum, *Modern operating systems*, 2^{nd} Edition, Prentice Hall (2001).