

## VERIFYING THE GOLDBACH CONJECTURE UP TO $4 \cdot 10^{14}$

JÖRG RICHSTEIN

ABSTRACT. Using a carefully optimized segmented sieve and an efficient checking algorithm, the Goldbach conjecture has been verified and is now known to be true up to  $4 \cdot 10^{14}$ . The program was distributed to various workstations. It kept track of maximal values of the smaller prime  $p$  in the minimal partition of the even numbers, where a minimal partition is a representation  $2n = p + q$  with  $2n - p'$  being composite for all  $p' < p$ . The maximal prime  $p$  needed in the considered interval was found to be 5569 and is needed for the partition  $389965026819938 = 5569 + 389965026814369$ .

### 1. INTRODUCTION

In his famous letter to Leonhard Euler dated 7 June 1742, Christian Goldbach first conjectures that every number that is a sum of two primes can be written as a sum of “as many primes as one wants”. Goldbach considered 1 as a prime and gives a few examples. In the margin of his letter, he then states his famous conjecture that every number is a sum of three primes. This is easily seen to be equivalent to that every even number is a sum of two primes which is referred to as the *Binary Goldbach Conjecture*. Its weaker form, the *Ternary Goldbach Conjecture* states that every odd number can be written as a sum of three primes. The ternary conjecture has been proved under the assumption of the truth of the generalized Riemann hypothesis ([5], [11]) and remains unproved unconditionally for only a finite (but not yet computationally coverable) set of numbers [4]. Although believed to be true, the binary Goldbach conjecture is still lacking a proof.

Two main approaches have been used in the past to efficiently verify the binary Goldbach conjecture: in order to verify its truth for a given interval  $[a, b]$ , one has to find sets of primes  $P_1$  and  $P_2$  such that

$$\{2n \mid a \leq 2n \leq b\} \subseteq P_1 + P_2 = \{p_1 + p_2 \mid p_1 \in P_1, p_2 \in P_2\}.$$

The difference between both ways is the choice and generation of the sets  $P_1$  and  $P_2$ . In method A,

$$P_1 = \{p \mid p \in [1, b - a + \delta]\}, \quad P_2 = \{p \mid p \in [a - \delta, a]\},$$

whereas in method B

$$P_1 = \{p \mid p \in [1, \delta]\}, \quad P_2 = \{p \mid p \in [a - \delta, b]\}.$$

Due to practical memory limitations, it will be necessary to partition  $[a, b]$  for large  $b - a$  and perform the generation of the set  $P_2$  for each of the resulting parts.

---

Received by the editor October 14, 1999 and, in revised form, January 6, 2000.  
2000 *Mathematics Subject Classification*. Primary 11P32; Secondary 11-04.  
*Key words and phrases*. Goldbach conjecture, distributed computing.

It turns out that in both methods, the value of  $\delta$  can be chosen to be very small compared to  $b$ . Therefore, the size of the sets  $P_2$  to be successively generated is much smaller in A than in B.

Method A was first implemented in [12]. As shown in [5], the generation of the small sets  $P_2$  can be carried out very efficiently by employing strong pseudoprime tests to several bases, which can currently be used to strictly prove primality up to 341550071728321 ([8]). This makes the first method preferable, since one needs to generate the primes from a much larger interval in method B.

Though method A is faster than method B, due to its current limitation and the fact that method A does not in general calculate the minimal partitions, the second method has been chosen here, implemented and distributed to numerous machines. Section 2 explains the subdivision of the interval  $[1, 4 \cdot 10^{14}]$  into “machine manageable” segments. The generation of the sets  $P_2$  is described in Section 3. In Section 4, the verification process is presented. Running times of the implementation and practical considerations are given in Section 5. Finally, the resulting maximal values of the minimal partitions are listed.

## 2. SEGMENTATION

Let  $\Pi_n$  denote the product of the first  $n$  odd primes. As a preparation for the distribution to several machines, the interval  $[1, 4 \cdot 10^{14}]$  was first split into subintervals  $I_k$  of length  $2^{26} \cdot \Pi_5 \approx 10^{12}$  (so  $0 \leq k \leq 396$ ). Each subinterval to be processed was further subdivided into segments  $S_k^i$  of length  $2^t \cdot \Pi_5$ , where the optimal value for  $t$  was determined experimentally depending on  $k$  (see Section 5). The segments  $S_k^i$  were mapped onto 32-bit integer arrays  $A_k^i$  of length  $2^{t-6} \cdot \Pi_5$  elements, such that a bit  $j$  ( $0 \leq j \leq 31$ ) of an array element  $A_k^i[m]$  ( $0 \leq i < 2^{26-t}, 0 \leq m < 2^{t-6} \cdot \Pi_5$ ) corresponds to the (odd) number  $2^{26} \cdot \Pi_5 \cdot k + 2^t \cdot \Pi_5 \cdot i + 2^6 \cdot m + 2 \cdot j + 1$ . As a preparation for the verification process, each array has been preceded with an additional  $\lceil \delta/2^6 \rceil$  number of elements  $A_k^i[-\lceil \delta/2^6 \rceil]$  through  $A_k^i[-1]$ . Initially,  $\delta$  was chosen to be equal to 10000.

## 3. GENERATION OF THE SETS $P_2$

In order to generate the sets  $P_2$  (to be represented by 1-bits in the arrays  $A_k^i$ ), an optimized, segmented version of Eratosthenes’ sieve as first used in [3] and later described in detail in [1] was implemented.

In addition to the basic method, the sieving of multiples of 3 and 5 that have already been sieved out has been avoided for prime factors between 17 and  $p_m$ , where  $p_m$  depends on the value of  $t$  (and therefore again on  $k$ ). The sieving process can be summarized as follows: the last  $\lceil \delta/2^6 \rceil$  elements of the previous segment  $A_k^{i-1}$  are copied onto the elements  $-\lceil \delta/2^6 \rceil$  through  $-1$  of  $A_k^i$ . The first segment  $S_0^0$  was handled separately; in the other cases where  $i = 0$ , the last segment  $S_{k-1}^{2^{26-t}-1}$  was processed again. Then, a prepared array with all bits set to 1 except for those which correspond to multiples of the first five odd primes is copied onto the current array  $A_k^i$  (starting with element 0). The factors  $l \cdot p$  of all primes  $17 \leq p < \sqrt{2^{26} \cdot \Pi_5 \cdot k + 2^t \cdot \Pi_5 \cdot (i+1)}$  to be sieved out are first adjusted to be congruent to  $2p \pmod{15}$  by successively adding  $2p$  to the first odd  $l \cdot p$  in the current segment and setting the corresponding bits of  $A_k^i$  to 0. Then, the sequence

$2p, 4p, 6p, 2p, 6p, 4p, 2p, 4p$  is repeatedly and successively added to the current multiple of  $p$  and the corresponding bits are set to 0 until the upper end of the segment is reached. This way, approximately 30% of the sieving costs were saved.

#### 4. VERIFICATION PROCESS

In order to find a partition for all even numbers of the current segment  $S_k^i$ , one can proceed as follows: A second array  $C$  of the same length as  $A_k^i$  (without the preceding  $\lceil \delta/2^6 \rceil$  elements) representing the even numbers of a segment  $S_k^i$  is initialized to all bits set to 0. Then, for each  $p_i \in P_1$ , the array  $A_k^i$  is shifted successively by  $(p_i - p_{i-1})/2$  bits starting with  $p_1 = 3$  (and  $p_0$  defined as 1). Now all bits of the shifted  $A_k^i$  correspond to even numbers  $q_1 + p_i, q_2 + p_i, \dots$  with  $q_l \in P_2$ .  $A_k^i$  is then joined to  $C$  by logical “or” (starting with  $A_k^i[0]$ ). By continuing this way, the segment  $S_k^i$  is completely verified when all bits of  $C$  are set to 1. Since it turns out that the array  $C$  fills up quickly, this addition of primes from  $P_1$  by shifting  $A_k^i$  will only be done up to some  $p_a$  and the remaining 0-bits handled separately (eventually leading to new maximal values of the minimal partitions).

The problem with this method is that it is relatively costly to perform the shiftings of  $A_k^i$  by variable  $(p_i - p_{i-1})/2$ . Additionally, it gets even more expensive when the difference between successive primes from  $P_1$  exceeds 64, because then additional shift/mask/add-operations will be necessary. Though the latter case will not be of relevance in the near future since a prime gap of 64 first occurs at 31397, the procedure can be simplified as follows, avoiding variable shifts. For each  $p \in P_1$ , the remainder  $(p - 1)/2 \pmod{32}$  is stored together with  $p$ . The whole array  $A_k^i$  is now shifted only 32 times by one bit (independent from the choice of  $p_a$ ). After the  $i$ th shifting (starting with  $i = 0$ ), all stored remainders  $(p - 1)/2 \pmod{32}$ ,  $p \in P_1$ ,  $p \leq p_a$  are checked for being equal to  $i$ . If so,  $C$  is or-joined to  $A_k^i$  shifted  $\lfloor (p - 1)/64 \rfloor$  words (where these word shifts are actually simple index changes). The last shift only adjusts  $A_k^i$ ; after the array has been shifted by a complete word, the copying of the last  $\lceil \delta/2^6 \rceil$  elements onto the next array  $A_k^{i+1}$  as mentioned in Section 3 then starts at element  $A_k^i[-\lceil \delta/2^6 \rceil + 1]$ . In order not to lose any maximal values of the minimal partitions, it is essential that  $p_a$  is always smaller than the previous maximum during the whole computation. Since the maximal  $p_a$  chosen was smaller than the maximal smaller prime already needed in the partitions of the even numbers of  $S_0^0$ , this problem was not of practical relevance.

#### 5. PRACTICAL CONSIDERATIONS AND RUNNING TIMES

Optimal values for the parameters  $t$ ,  $p_m$  and  $p_a$  have been precalculated experimentally in intervals of length  $10^{13}$  on each machine type involved. Depending on  $k$ , sources were rebuilt and recompiled with a new parameter combination. It turned out that the ranges  $9 \leq t \leq 11$ ,  $40000 \leq p_m \leq 120000$  and  $181 \leq p_a \leq 307$  yielded the best results. The total memory requirement was therefore at most 4MB for the arrays  $A_k^i$  and  $C$ .

The program was written in C and distributed to seven Sun Ultra1 and six Sun4 workstations. In addition, two PC's running Linux have been used. Approximately  $10^7$  even numbers were checked per second. The time until completion was around 130 days with processes running on lowest priority in background. The share of the costs to generate the sets  $P_2$  varied between 45% at  $10^{12}$  and 60% near  $4 \cdot 10^{14}$ .

TABLE 1.

$n$	$p(n)$	$\frac{p(n)}{\log^2 n \log \log n}$	$n$	$p(n)$	$\frac{p(n)}{\log^2 n \log \log n}$
6	3	1.6023	113632822	1163	1.1575
12	5	0.8896	187852862	1321	1.2350
30	7	0.4943	335070838	1427	1.2440
98	19	0.5935	419911924	1583	1.3436
220	23	0.4691	721013438	1789	1.4262
308	31	0.5408	1847133842	1861	1.3357
556	47	0.6380	7473202036	1877	1.1625
992	73	0.7939	11001080372	1879	1.1191
2642	103	0.8037	12703943222	2029	1.1912
5372	139	0.8762	21248558888	2089	1.1658
7426	173	0.9956	35884080836	2803	1.4873
43532	211	0.7808	105963812462	3061	1.4686
54244	233	0.8207	244885595672	3163	1.4080
63274	293	0.9977	599533546358	3457	1.4243
113672	313	0.9410	3132059294006	3463	1.2452
128168	331	0.9708	3620821173302	3529	1.2543
194428	359	0.9685	4438327672994	3613	1.2637
194470	383	1.0332	5320503815888	3769	1.2996
413572	389	0.9086	8342945544436	3917	1.3042
503222	523	1.1784	10591605900482	4003	1.3086
1077422	601	1.1839	12982270197518	4027	1.2962
3526958	727	1.1790	15197900994218	4057	1.2903
3807404	751	1.2034	28998050650046	4327	1.3114
10759922	829	1.1357	46878442766282	4519	1.3221
24106882	929	1.1349	76903574497118	4909	1.3859
27789878	997	1.1943	184162477860248	5077	1.3476
37998938	1039	1.1928	217361316706568	5209	1.3668
60119912	1093	1.1807	389965026819938	5569	1.4038

## 6. RESULTS

In [7], it was conjectured that  $p(n) \ll \log^2 n \log \log n$ . In the range covered there, it was derived that  $p(n) < 1.603 \cdot \log^2 n \log \log n$ . As an extension to results from [2], [7] and [13], Table 1 shows the maximal values of the function  $p(n) = \min\{p \mid \exists q : n = p + q\}$  for even  $n$  along with the quotients  $p(n)/\log^2 n \log \log n$ .

In [2], it was suggested that the quotient  $\log n/\log^2 p(n)$  stays approximately constant. Contrary to that, it seems likely from the data obtained that the quotient tends to infinity with  $n$ . Note that this also follows from the conjecture  $p(n) \ll \log^2 n \log \log n$  of [7].

## 7. REMARK

An exception to the ternary Goldbach conjecture would now require a prime gap of  $4 \cdot 10^{14}$  below  $10^{43000}$ , from whereon it is known to be true unconditionally ([4]). Since a gap of that size is expected to occur first at around  $10^{14000000}$  ([10]),

the existence of a counterexample to the ternary conjecture has become even more unlikely.

## 8. ACKNOWLEDGMENTS

The author wishes to thank Andrew Granville for his helpful comments and suggestions, and the staff of the Institut für Informatik, Universität Giessen, for their support and the computing time necessary to carry out this work.

## REFERENCES

1. C. Bays, R. Hudson, *The Segmented Sieve of Eratosthenes and Primes in Arithmetic Progressions*, BIT **17** (1977), 121–127. MR **56**:5405
2. J. Bohman, C. E. Fröberg, *Numerical Results on the Goldbach Conjecture*, BIT **15** (1975), 239–243. MR **52**:10644
3. R. P. Brent, *The first occurrence of large gaps between successive primes*, Math. Comp., **27** (1973), 959–963. MR **48**:8360
4. J. R. Chen, Y. Wang, *On the odd Goldbach problem*, Acta Math. Sinica, **32** (1989), 702–718. MR **91e**:11108
5. J. M. Deshouillers, H. J. J. te Riele, Y. Saouter *New experimental results concerning the Goldbach conjecture*, Proc. 3rd Int. Symp. on Algorithmic Number Theory, LNCS **1423** (1998), 204–215. CMP 2000:05
6. P. H. Fuss, *Correspondance mathématique et physique de quelques célèbres géomètres du XVIII<sup>e</sup> siècle*, tome I, St. Pétersbourg, (1843), 127+135.
7. A. Granville, H. J. J. te Riele, J. van de Lune *Checking the Goldbach Conjecture on a Vector Computer*, Number Theory and Applications (R. A. Mollin, ed.), Kluwer, Dordrecht (1989), 423–433. MR **93c**:11085
8. G. Jaeschke, *On strong pseudoprimes to several bases*, Math. Comp. **61** (1993), 915–926. MR **94d**:11004
9. A. P. Juškevič, *Christian Goldbach: 1690–1764*, Vita mathematica, Birkhäuser Basel (1994), 161.
10. H. Riesel, *Prime Numbers and Computer Methods for Factorization*, 2nd Edition, Birkhäuser, (1994). MR **95h**:11142
11. Y. Saouter, *Checking the odd Goldbach conjecture up to  $10^{20}$* , Math. Comp. **67** (1998), 863–866. MR **98g**:11115
12. M. K. Shen, *On checking the Goldbach Conjecture*, BIT **4** (1964), 243–245. MR **30**:3051
13. M. K. Sinisalo, *Checking the Goldbach Conjecture up to  $4 \cdot 10^{11}$* , Math. Comp. **61** (1993), 931–934. MR **94a**:11157
14. M. L. Stein, P. R. Stein, *Experimental results on additive 2 bases*, BIT **38** (1965), 427–434.

INSTITUT FÜR INFORMATIK, FACHBEREICH MATHEMATIK, JUSTUS-LIEBIG-UNIVERSITÄT, GIESSEN, GERMANY

*E-mail address:* Joerg.Richstein@informatik.uni-giessen.de