MATHEMATICS OF COMPUTATION
Volume 66, Number 220, October 1997, Pages 1709–1717
S 0025-5718(97)00864-8

FURTHER TABULATION OF THE ERDÖS-SELFRIDGE FUNCTION

RICHARD F. LUKES, RENATE SCHEIDLER, AND HUGH C. WILLIAMS

ABSTRACT. For a positive integer k, the Erdös-Selfridge function is the least integer $g(k) > k+1$ such that all prime factors of $\binom{g(k)}{k}$ exceed k. This paper describes a rapid method of tabulating $g(k)$ using VLSI based sieving hardware. We investigate the number of admissible residues for each modulus in the underlying sieving problem and relate this number to the size of $g(k)$. A table of values of $g(k)$ for $135 \le k \le 200$ is provided.

1. INTRODUCTION

For $k \ge 1$, denote by $g(k)$ the least integer $> k+1$ such that no prime $p \le k$ divides $\binom{g(k)}{k}$. This function grows rapidly with increasing k and is consequently difficult to compute for even modest values of k. The behavior of $g(k)$ was first studied by Ecklund, Erdös and Selfridge [2] who tabulated $g(k)$ for $k \le 40$ as well as $g(42)$, $g(46)$, and $g(52)$. These are all the values of $g(k) \le 2500000$ when $k \le 100$. The table was extended to include all the values of $g(k)$ for $k \le 140$ by Scheidler and Williams [1] using sieving techniques. The largest of these values, $g(139)$, is a 17 digit number. Sieving was continued for $141 \le k \le 155$ but the results were never published.

A number of lower bounds on $g(k)$ were proved and conjectured in [2] and by Erdös, Lacampagne and Selfridge in [3]. The best lower bound was recently established by Granville and Ramaré [5] who proved that there exists an absolute positive constant c such that

$$g(k) > \exp(c(\log^3 k/\log\log k)^{\frac{1}{2}}).$$

This implies that $g(k)$ grows faster than any polynomial in k.

This paper further extends computations and provides values of $g(k)$ for $135 \le k \le 200$. We also repeated earlier tabulations and found an error in the value of $g(138)$ given in [1]. The computation was performed on the *Manitoba Scalable Sieve Unit* (MSSU), a very fast VLSI based sieving device developed by Lukes, Patterson and Williams [4]. We used a modification of the algorithm given in [1]. To make this paper somewhat self-contained, we begin with a brief review of the basics of sieving as well as the sieving method used in our computations. We analyze the number of admissible residues of the sieving problem arising from $g(k)$ in Section 3. Section 4 compares the size of the sieving problem for $g(k)$ with the actual value

Received by the editor October 18, 1994 and, in revised form, October 9, 1995 and August 21, 1996.

1991 *Mathematics Subject Classification.* Primary 11N25, 11Y70, 11-04.

The third author's research is supported by NSERC of Canada grant A7649.

of $g(k)$ and investigates gaps between $g(q-1)$ and $g(q)$ where q is a prime. Our implementation on MSSU is discussed in Section 5. The paper concludes with a table of values of $g(k)$ $(135 \leq k \leq 200)$.

2. The sieving algorithm

In order to solve a *sieving problem*, it is required to find solutions to a system of simultaneous linear congruences. More exactly, one needs to search for integers x such that

$$x \pmod{m_i} \in R_i \qquad \text{for} \quad i = 1, 2, \ldots, h, \tag{2.1}$$

where h is a positive integer, the moduli $m_1, m_2, \ldots, m_h$ are positive integers assumed to be pairwise relatively prime, and each set $R_i = \{r_{i1}, r_{i2}, \ldots, r_{in_i}\}$, called the set of *admissible residues* for modulus m_i, consists of nonnegative integers less than m_i $(i = 1, 2, \ldots, h)$. Boundary conditions are placed on x, i. e. we may require x to lie in a certain specified range, or we might wish to obtain the least solution of (2.1) that exceeds a fixed lower bound. Additional restrictions, checked by a *filter*, may be placed on x.

Over the past 75 years, a number of mechanical as well as computer based machines for solving sieving problems have been constructed (see [4] for a history of these machines). MSSU is the most recent and by far the fastest such device.

Our method for tabulating $g(k)$ as well as some of the results in Section 3 are derived from Kummer's well-known result that the binomial coefficient $\binom{n}{k}$ is relatively prime to a prime p if and only if there are no "carries" when k and $n-k$ are added in base p (see [6, p. 220]). Since the sieving algorithm is described in detail in [1], we merely sketch it here. Let

$$k = \sum_{i=0}^{m} a_i p^i, \qquad 0 \leq a_i \leq p-1 \quad \text{for} \quad i = 0, 1, \ldots, m; \quad a_m \neq 0$$

be the base p representation of k. (It is easy to compute the coefficients a_i, see formula (2.1) in [1]). For $i = 0, 1, \ldots, m$, set

$$C_i = \{a_i, a_i + 1, \ldots, p-1\}$$

and recursively define the sets

$$B_0 = C_0, \quad B_i = B_{i-1} + C_i p^i = \{b + cp^i \mid b \in B_{i-1}, c \in C_i\} \quad (i = 1, 2, \ldots, m).$$

Then $B_i = C_0 + C_1 p + C_2 p^2 + \cdots + C_i p^i$ for $i = 0, 1, \ldots, m$, and $g(k)$ is the smallest integer $n \geq k+2$ such that

$$n \pmod{p^{m+1}} \in B_m \tag{2.2}$$

or equivalently,

$$n \pmod{p^m} \in B_{m-1} \tag{2.3}$$

and

$$\left\lfloor \frac{n}{p^m} \right\rfloor \pmod{p} \geq a_m. \tag{2.4}$$

To compute $g(k)$ for fixed k, the moduli m_i in (2.1) are the values p^m where p is a prime, $p \leq k$, and $p^m \leq k < p^{m+1}$, i. e. $m = \lfloor \log_p k \rfloor$. For each modulus p^m, the corresponding set of admissible residues is B_{m-1}. Each solution n of (2.3) is checked for filter condition (2.4). The least integer $n \geq k+2$ satisfying both (2.3) and (2.4) for all primes $p \leq k$ is $g(k)$.

3. Number of admissible residues

Let $p \leq k$ be a fixed prime and set $m = \lfloor \log_p k \rfloor$. Then each set C_i contains $|C_i| = p - a_i$ elements ($i = 0, 1, \ldots, m$), so for $i \neq j$, we have $|C_i p^i + C_j p^j| = (p-a_i)(p-a_j)$. Hence, the number of residues for each modulus is given as follows.

Lemma 3.1. *The number of admissible residues for modulus p^m is*

$$r_p = |B_{m-1}| = \prod_{i=0}^{m-1} (p - a_i).$$

If the modulus is a prime, i. e. $m = 1$, then the base p representation of k is $k = a_1 p + a_0$, so $p - a_0 = (a_1 + 1)p - k$. Hence in this case, we have

Corollary 3.2. *If the modulus is a prime p, then p divides $k + r_p$.*

Clearly, the number of residues r_p for modulus p^m is between 1 and p^m, inclusive. If the number of admissible residues is maximal, i. e. $r_p = p^m$, then (2.3) is always satisfied and we do not need to include modulus p^m in the congruences. It is easy to establish the exact form of k in the extreme cases $r_p = p^m$ and $r_p = 1$. For single residue congruences, we can also determine the unique residue.

Lemma 3.3. *$r_p = p^m$ if and only if $k = ap^m$ where $1 \leq a \leq p-1$.*

Proof. By Lemma 3.1, $r_p = p^m$ if and only if $a_i = 0$ for $i = 0, 1, \ldots m-1$, so $k = a_m p^m$, $1 \leq a_m \leq p-1$. □

Lemma 3.4. *$r_p = 1$ if and only if $k = ap^m - 1$ where $2 \leq a \leq p$.*

Proof. By Lemma 3.1, $r_p = 1$ if and only if $a_i = p-1$ for $i = 0, 1, \ldots, m-1$, so

$$k = a_m p^m + \sum_{i=0}^{m-1} (p-1)p^i = a_m p^m + p^m - 1 = (a_m + 1)p^m - 1,$$

$1 \leq a_m + 1 \leq p$. □

Lemma 3.5. *If $r_p = 1$, then the residue corresponding to modulus p^m is $p^m - 1$.*

Proof. As in the previous lemma, if $r_p = 1$, then $a_i = p-1$ for $i = 0, 1, \ldots m-1$, i. e. B_{m-1} contains only the residue $\sum_{i=0}^{m-1} (p-1)p^i = p^m - 1$. □

We now compare the number of admissible residues for the two consecutive values $g(k-1)$ and $g(k)$ in the special case where k is a prime power. Let $k = q^t$ where q is a prime and $t \geq 1$. As before, let $p \leq k$ be a fixed prime and set $m = \lfloor \log_p k \rfloor$. To distinguish between quantities pertaining to different k values, we include k as an argument, i. e. write $r_p(k)$, $C_i(k)$ etc.

Case 1: $p = q$. Then $m = t$, $k = p^m$ and by Lemma 3.3, $r_p(k) = p^m$. Now $k - 1 = p^m - 1$, so if $m > 1$, then $k - 1 = p \cdot p^{m-1} - 1$ and $r_p(k-1) = 1$ by Lemma 3.4 (here, the corresponding modulus is p^{m-1}). If $m = 1$, then $k - 1 = p - 1 < p$, so the moduli used in searching for $g(k-1)$ do not include a power of p.

Case 2: $p \neq q$. Then it is easy to see that $\lfloor \log_p(k-1) \rfloor = m$. Let the p-ary representation of $k-1$ be

$$k - 1 = \sum_{i=0}^{m} a_i p^i, \qquad 0 \le a_i \le p-1 \quad \text{for} \quad i = 0, 1, \dots, m; \quad a_m \neq 0.$$

Since p does not divide $k = q^t$, we must have $a_0 \neq p-1$, so the p-ary representation of k is

$$k = \sum_{i=1}^{m} a_i p^i + (a_0 + 1).$$

Hence $C_0(k) = C_0(k-1) \setminus \{a_0\}$ and $C_i(k) = C_i(k-1)$ for $i = 1, 2, \dots m$. It follows that

$$\begin{aligned} r_p(k) &= (p - a_0 - 1) \prod_{i=1}^{m-1} (p - a_i) = r_p(k-1) - \prod_{i=1}^{m-1} (p - a_i) \\ &= r_p(k-1) \left(1 - \frac{1}{p - a_0} \right). \end{aligned}$$

In summary:

Lemma 3.6. *Let $k = q^t$, q a prime, $t \ge 1$. Then for any prime $p \le k$, the number of admissible residues for modulus p^m, $m = \lfloor \log_p k \rfloor$, satisfies the following properties.*

1. *If $p = q$, then $r_p(k) = p^m$, $r_p(k-1) = 1$ if $m > 1$, and no power of p is included in the moduli for $k-1$ if $m = 1$.*
2. *If $p \neq q$, then*

$$r_p(k) = r_p(k-1) \left(1 - \frac{1}{p - k_p} \right),$$

where $k_p \equiv k - 1 \pmod{p}$, $1 \le k_p \le p-1$.

Corollary 3.7. *Let $k = q$ be a prime. Then the moduli in the congruences for both $g(q)$ and $g(q-1)$ are exactly the powers p^m where p is a prime less than q and $m = \lfloor \log_p k \rfloor$. Furthermore, for each such prime p,*

$$r_p(q) = r_p(q-1) \left(1 - \frac{1}{p - q_p} \right),$$

where $q_p \equiv q - 1 \pmod{p}$, $1 \le q_p \le p-1$.

We conclude this section with a brief analysis of the filter conditions for both k and $k-1$ when $k = q$ is a prime.

Lemma 3.8. *If $k = q$ is a prime, then the filter condition (2.4) is satisfied for any solution candidate n for either $g(q)$ or $g(q-1)$.*

Proof. Let $p \le q$ be a prime. Since each solution candidate n for either $g(q)$ or $g(q-1)$ satisfies $n \ge (q-1) + 2 = q + 1 > p$, the left-hand side of (2.4) is always at least 1. If $p < q$, then the base p representations of q and $q-1$, respectively, are $q = p + (q-p)$ and $q - 1 = p + (q - p - 1)$, so in either case $a_m = a_1 = 1$ and (2.4) always holds. If $p = q$, then the prime p is not included in the filter condition for $k = q-1$, and for $k = q$, we have again $a_m = a_1 = 1$, so (2.4) is always satisfied. □

is combined into a single congruence

$$x \equiv S_1, S_2, \dots, S_l \pmod{M} \tag{5.2}$$

where $M = m_1 m_2 \cdots m_g$ and the S_i, $(i = 1, 2, \dots, l)$ are obtained using the Chinese Remainder Theorem. If the set of congruences (5.1) is selected such that l in (5.2) satisfies $l \leq 32$, then each residue S_i in (5.2) can be assigned to a different sieve chip. The i-th chip now sieves on $Mx + S_i$ rather than x, which results in a speed-up of a factor M in the computation. Therefore, the congruences (5.1) should be selected among the many different partitions in such a way that M is maximal. Clearly, congruences with few admissible residues (single residue congruences in particular) and large moduli are most desirable. The number of choices is further increased when partitioning is combined with residue folding, since for partitioning purposes, a reduced modulus $m_j = p^l$ in (5.1) need not actually be supported by the underlying hardware. Fortunately, the total number of congruences is sufficiently small to make an exhaustive search for the optimal combination of residue folding and congruence partitioning computationally feasible.

We conclude this section with a comment on the speed-up suggested in [1] in the case where $k+1$ is composite. In this case, each solution candidate n for $g(k)$ satisfies $n \equiv -1 \pmod{k+1}$, so one can sieve on $(n+1)/(k+1)$ rather than n and speed up the process by a factor of $k+1$. The MSSU algorithm achieves essentially the same speed-up as follows. For each prime divisor p of $k+1$, modulus p^m is folded onto modulus p^α where α is the largest exponent such that $p^\alpha \mid k+1$. This results in a single residue congruence $n \equiv -1 \pmod{p^\alpha}$ (see the proof of Lemma 2 in [1]). Combining these congruences for all primes dividing $k+1$ yields a single residue congruence $n \equiv -1 \equiv k \pmod{k+1}$.

We recomputed $g(k)$ for all $k \leq 140$ and found an error in the table given in [1] for $k = 138$. The correct value is $g(k) = 601242167764223$. We also computed $g(k)$ for $141 \leq k \leq 200$. A table of these values can be found at the end of the paper. To show the enormous increase in speed of MSSU versus OASiS, the device used for the computations in [1], we point out that OASiS required 11 days 11 hours for computing $g(139)$, whereas MSSU achieved this task in a mere 4 minutes (including time to load and verify the problem).

Sieving rates varied greatly for various values of k. The fastest sieving rate occurred for $k = 199$ with a hardware count rate of 7.5×10^{15} per second, requiring less than 20 hours to compute $g(199)$. One of the more difficult values of k to compute was 198 with a hardware sieving rate of 3.3×10^{13}. This would have taken more than 50 days to compute using only 32 sieve chips. However, we were able to re-partition the problem into 5 subproblems requiring 24 sieve chips each and were able to verify a solution in under 10 days. Due to the very low rate at which solution candidates were generated, solution filtering had a negligible effect on the the observed sieving rate. Surprisingly, even with the introduction of false solutions by residue folding, and optimizing out many of the sparse congruences using partitioning, sieving proceeded at essentially the maximum theoretical hardware sieving rate. Using a 6-way partitioning, it took approximately 30 days to compute the largest value found, $g(200)$, which is a 23 digit number.

In the table of values of $g(k)$, the digits of $g(k)$ are written in groups of at most ten to facilitate reading. Prime values of k are given in bold type.

The authors wish to thank the referee for several helpful suggestions.

Lemma 3.8, we can ignore the filter condition (2.4), so from (2.3), we obtain

$$P_p(k) = \frac{r_p(k)}{p^m}$$

for both $k = q$ and $k = q + 1$. Corollary 3.7 implies that

$$r_p(q) \le r_p(q-1)\left(1 - \frac{1}{p}\right),$$

therefore

$$P(q) \le P(q-1) \prod_{p \le q-1} \left(1 - \frac{1}{p}\right).$$

By Mertens' theorem

$$\prod_{p \le k} \left(1 - \frac{1}{p}\right) \sim \frac{e^{-\gamma}}{\log k},$$

where γ is Euler's constant, hence the two probabilities $P(q-1)$ and $P(q)$ will tend to differ by at least a factor which is proportional to $\log q$. Thus, we expect the gaps between $g(q-1)$ and $g(q)$ to increase significantly for large primes q.

5. Implementation

MSSU utilizes 32 VLSI chips operating in parallel, each of which implements an electronic sieve device performing at a rate of 192 million trials per second. Each individual chip supports the moduli 16, 9, 25, and 49, as well as the next 26 primes 53 through 113 in hardware.

For fixed k, the values of all the admissible residues in the sieving problem for $g(k)$ are precomputed in software and passed to MSSU. MSSU then optimizes this information as described below to best fit its hardware. An on-line filter checks each value n which satisfies (2.3) for condition (2.4). The computation terminates as soon as such a value n is let through by the filter, this value being $g(k)$.

Since many of the required moduli p^m are not available in hardware, a congruence $\pmod{p^m}$ may be reduced to a congruence $\pmod{p^l}$ where $l < m$. The residues in B_{m-1} are then mapped or *folded* onto a possibly smaller set of residues $\pmod{p^l}$. The congruences "lost" in the process of residue folding are implemented in software using an off-line filter that screens out "false" solutions. This does not slow down the sieving process, as the number of false solutions is sufficiently small to avoid a bottleneck.

MSSU further optimizes the computation by *partitioning* congruences. A subset of moduli $\{m_1, m_2, \dots, m_g\}$ is selected, and for each modulus m_j, a subset $\{s_{j1}, s_{j2}, \dots, s_{jl_j}\}$ of the corresponding admissible residues is chosen ($j = 1, 2, \dots, g$). The set of congruences

$$x \equiv s_{j1}, s_{j2}, \dots, s_{jl_j} \pmod{m_j} \quad (j = 1, 2, \dots, g) \tag{5.1}$$

is combined into a single congruence

$$x \equiv S_1, S_2, \ldots, S_l \pmod{M} \tag{5.2}$$

where $M = m_1 m_2 \cdots m_g$ and the S_i, $(i = 1, 2, \ldots, l)$ are obtained using the Chinese Remainder Theorem. If the set of congruences (5.1) is selected such that l in (5.2) satisfies $l \leq 32$, then each residue S_i in (5.2) can be assigned to a different sieve chip. The i-th chip now sieves on $Mx + S_i$ rather than x, which results in a speed-up of a factor M in the computation. Therefore, the congruences (5.1) should be selected among the many different partitions in such a way that M is maximal. Clearly, congruences with few admissible residues (single residue congruences in particular) and large moduli are most desirable. The number of choices is further increased when partitioning is combined with residue folding, since for partitioning purposes, a reduced modulus $m_j = p^l$ in (5.1) need not actually be supported by the underlying hardware. Fortunately, the total number of congruences is sufficiently small to make an exhaustive search for the optimal combination of residue folding and congruence partitioning computationally feasible.

We conclude this section with a comment on the speed-up suggested in [1] in the case where $k+1$ is composite. In this case, each solution candidate n for $g(k)$ satisfies $n \equiv -1 \pmod{k+1}$, so one can sieve on $(n+1)/(k+1)$ rather than n and speed up the process by a factor of $k+1$. The MSSU algorithm achieves essentially the same speed-up as follows. For each prime divisor p of $k+1$, modulus p^m is folded onto modulus p^α where α is the largest exponent such that $p^\alpha \mid k+1$. This results in a single residue congruence $n \equiv -1 \pmod{p^\alpha}$ (see the proof of Lemma 2 in [1]). Combining these congruences for all primes dividing $k+1$ yields a single residue congruence $n \equiv -1 \equiv k \pmod{k+1}$.

We recomputed $g(k)$ for all $k \leq 140$ and found an error in the table given in [1] for $k = 138$. The correct value is $g(k) = 601242167764223$. We also computed $g(k)$ for $141 \leq k \leq 200$. A table of these values can be found at the end of the paper. To show the enormous increase in speed of MSSU versus OASiS, the device used for the computations in [1], we point out that OASiS required 11 days 11 hours for computing $g(139)$, whereas MSSU achieved this task in a mere 4 minutes (including time to load and verify the problem).

Sieving rates varied greatly for various values of k. The fastest sieving rate occurred for $k = 199$ with a hardware count rate of 7.5×10^{15} per second, requiring less than 20 hours to compute $g(199)$. One of the more difficult values of k to compute was 198 with a hardware sieving rate of 3.3×10^{13}. This would have taken more than 50 days to compute using only 32 sieve chips. However, we were able to re-partition the problem into 5 subproblems requiring 24 sieve chips each and were able to verify a solution in under 10 days. Due to the very low rate at which solution candidates were generated, solution filtering had a negligible effect on the the observed sieving rate. Surprisingly, even with the introduction of false solutions by residue folding, and optimizing out many of the sparse congruences using partitioning, sieving proceeded at essentially the maximum theoretical hardware sieving rate. Using a 6-way partitioning, it took approximately 30 days to compute the largest value found, $g(200)$, which is a 23 digit number.

In the table of values of $g(k)$, the digits of $g(k)$ are written in groups of at most ten to facilitate reading. Prime values of k are given in bold type.

The authors wish to thank the referee for several helpful suggestions.

k	$g(k)$
135	315 7756005623
136	413 8898693368
137	95159 8054985213
138	60124 2167764223
139	2597202 7636644319
140	908985 4222866845
141	6333152 3816662671
142	1990465 6320115423
143	1542289 5461804543
144	139719 3586455769
145	339515 6674599871
146	17509 5016485374
147	72531 1731192223
148	1180 8400809148
149	542394 5342959799
150	47313 8520098551
151	3258989 9217872863
152	15549796 7465547419
153	53518499 5256751839
154	17864690 7528990874
155	4129798 4000013467
156	2527233 4970944959
157	104130829 7102375167
158	4702566 0758882783
159	6485551 8266246559
160	1664745 6280932287
161	342159 0108339941
162	20212 9337635322
163	5188127 2225707439
164	14664726 1829992439
165	9865227 4401898671
166	347584 7868933047
167	277308556 4165092343
168	48669223 2365306798
169	72698097 9380669099

k	$g(k)$
170	5942841 5007516671
171	28496594 9074228671
172	11223206 5794463997
173	138175311 6390427373
174	11057733 6695616174
175	194409219 4361247743
176	22625530 3912072703
177	19246523 8561441207
178	684480 9280136434
179	90787419 7930300859
180	43976301 6255983614
181	2 8336150170 1232528573
182	2898883863 4918997183
183	5351624705 6143575999
184	2662687844 8827721469
185	3713603655 0263266493
186	1 4274157994 6200597438
187	220884991 2824359867
188	127198106 5611178943
189	295629805 3153332989
190	45565223 2192890367
191	939688321 4719852991
192	106365072 4436901873
193	4 3525141972 8230720249
194	1 2638743588 3753706219
195	2632591216 1870817495
196	78151666 4215365373
197	4 2796097712 6350089949
198	1 3533936460 3654686198
199	4 0316886886 7096129999
200	520 8783889271 0191382732

References

1. R. Scheidler and H. C. Williams, *A method of tabulating the number theoretic function $g(k)$*, Math. Comp. **59** (1992), 251–257. MR **92k:**11146
2. E. F. Ecklund, P. Erdös and J. L. Selfridge, *A new function associated with the prime factors of* $\binom{n}{k}$, Math. Comp. **28** (1974), 647–649. MR **49:**2501
3. P. Erdös, C. B. Lacampagne and J. L. Selfridge, *Estimates of the least prime factor of a binomial coefficient*, Math. Comp. **61** (1993), 215–224. MR **93k:**11013
4. R. F. Lukes, C. D. Patterson and H. C. Williams, *Numerical Sieving Devices: Their History and Some Applications*, Nieuw Archiv voor Wiskunde **13**, ser. 4, no. 1 (1995), 113–139. MR **96m:**11082

5. A. Granville and O. Ramaré, *Explicit bounds on exponential sums and the scarcity of square-free binomial coefficients*, Mathematika, **43** (1996), 73–107. CMP 96:16
6. L. E. Dickson, *History of the Theory of Numbers*, vol. 1, Chelsea, New York, 1966. MR **39**:6807a

Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada R3T 2N2
E-mail address: rflukes@cs.umanitoba.ca

Department of Mathematical Sciences, University of Delaware, Newark, Delaware 19716
E-mail address: scheidle@math.udel.edu

Department of Computer Science, University of Manitoba, Winnipeg, Manitoba, Canada R3T 2N2
E-mail address: hugh_williams@csmail.cs.umanitoba.ca