MATHEMATICS OF COMPUTATION
Volume 66, Number 218, April 1997, Pages 893–901
S 0025-5718(97)00818-1

DECOMPOSING 40 BILLION INTEGERS BY FOUR TETRAHEDRAL NUMBERS

CHUNG-CHIANG CHOU AND YUEFAN DENG

ABSTRACT. Based upon a computer search performed on a massively parallel supercomputer, we found that any integer n less than 40 billion (40B) but greater than $343,867$ can be written as a sum of four or fewer tetrahedral numbers. This result has established a new upper bound for a conjecture compared to an older one, 1B, obtained a year earlier. It also gives more accurate asymptotic forms for partitioning.

All this improvement is a direct result of algorithmic advances in efficient memory and cpu utilizations. The heuristic complexity of the new algorithm is $O(n)$ compared with that of the old, $O(n^{5/3}\log n)$.

1. INTRODUCTION

Both papers [1] and [2] have demonstrated the partitioning of integers into tetrahedral numbers defined by

$$T(m) = (m-1)m(m+1)/6, \text{ where } m > 1, \tag{1}$$

by means of computation. We denote a number n as a k-number if n is a sum of k tetrahedral numbers and is not a sum of fewer than k tetrahedral numbers.

It can be shown with an explicit form of the circle method that all sufficiently large integers may be expressed as the sum of at most seven tetrahedral numbers. In [1], Deng and Yang reported that any integer satisfying $343,867 < n \leq 1\text{B}$ can be written as a sum of four or fewer tetrahedral numbers based upon a search on a distributed-memory parallel computer. That paper also addressed the main issues in numerical study of the Waring problem dealing with the tetrahedral numbers. The algorithm in [1] costs $O(n^{4/3}\log n)$ for searching all 3-numbers and $O(n^{5/3}\log n)$ for 4-numbers. We have improved that algorithm; it is now perfectly load balanced and costs $O(n)$ for all 3-numbers and 4-numbers.

The main purpose of the present paper is to describe the fast search algorithm and the decomposition of 40 billion (40B) integers by four tetrahedral numbers. In §2, we report the decomposition results obtained on an Intel Paragon. In §3, we discuss the asymptotic distribution of the partition. In §4, we describe and analyze the algorithms in both sequential and parallel forms. The conclusion is given in §5.

Received by the editor February 20, 1995 and, in revised form, May 22, 1995 and March 27, 1996.

1991 *Mathematics Subject Classification.* Primary 11P05, 65Y05, 68Q25.

Key words and phrases. Waring's problem, parallel computing, asymptotic form.

2. Partitioning

Our main results concern $N_k(n)$, defined as the number of k-numbers in the interval $[1, n]$, and are tabulated in Table 1. They show that among all positive integers up to 40B, there are 241 that require five tetrahedral numbers to decompose while the rest of the integers require only four or fewer numbers. Summarizing the results, we obtain the following two theorems:

Theorem 1. *Any integer n_4, satisfying*

$$343,867 < n_4 \leq 40\text{B}$$

can be written as a sum of four or fewer tetrahedral numbers.

Proof. It is shown by the computer search results. □

Theorem 2. *Any positive integer less than $T(L) = 3,771,207,667,368,141$ can be written as a sum of at most five tetrahedral numbers, where $L = 282,842$ is the largest integer for which*

$$T(L-1) - T(L-2) + 343,867 < 40\text{B}.$$

Proof. We notice that $T(6,214) < 40\text{B}$. Thus we only need to show that all integers are sums of at most five tetrahedral numbers between $T(6,214)$ and $T(282,842)$. Between $T(m)$ and $T(m+1)$ in the interval, we can divide all integers n into an upper group for which

$$T(m) + 343,867 < n \leq T(m+1),$$

and a lower group for which

$$T(m) < n \leq T(m) + 343,867.$$

For an integer n in the upper group, $n - T(m)$ satisfies

$$\begin{aligned} 343,867 \quad &< n - T(m) \\ &\leq T(m+1) - T(m) \\ &\leq T(L) - T(L-1) \\ &< 40\text{B}. \end{aligned}$$

By Theorem 1, $n - T(m)$ is a sum of at most four tetrahedral numbers. Thus any upper group integer can be written as a sum of at most five tetrahedral numbers.

For an integer n in the lower group, n satisfies

$$\begin{aligned} 343,867 \quad &< T(6,214) - T(6,213) \\ &\leq T(m) - T(m-1) \\ &< n - T(m-1) \\ &\leq T(m) - T(m-1) + 343,867 \\ &\leq T(L-1) - T(L-2) + 343,867 \\ &< 40\text{B}. \end{aligned}$$

Thus any lower group integer can be written as a sum of at most five tetrahedral numbers.

Therefore n is expressible as a sum of at most five tetrahedral numbers. □

TABLE 1. This table shows the partitioning of integers in intervals of [1, 40B] into 1-, 2-, 3-, and 4-numbers. The count for 5-numbers in the interval is always 241

n	$N_1(n)$	$N_2(n)$	$N_3(n)$	$N_4(n)$
1B	1816	1451433	446186613	552359897
2B	2288	2305850	892371789	1105319832
3B	2619	3022708	1338554381	1658420051
4B	2883	3662773	1784740032	2211594071
5B	3106	4251139	2230917514	2764828000
6B	3300	4801163	2677128667	3318066629
7B	3475	5321470	3123320579	3871354235
8B	3633	5817556	3569522288	4424656282
9B	3778	6293201	4015712155	4977990625
10B	3913	6751629	4461907459	5531336758
11B	4040	7195117	4908109379	6084691223
12B	4159	7625308	5354324211	6638046081
13B	4271	8043637	5800524169	7191427682
14B	4378	8451444	6246745411	7744798526
15B	4480	8849600	6692919707	8298225972
16B	4577	9238954	7139115585	8851640643
17B	4671	9620463	7585316341	9405058284
18B	4761	9994496	8031509522	9958490980
19B	4847	10361548	8477729794	10511903570
20B	4931	10722454	8923960648	11065311726
21B	5012	11077314	9370195643	11618721790
22B	5090	11426471	9816423186	12172145012
23B	5166	11770443	10262650165	12725573985
24B	5240	12109499	10708861927	13279023093
25B	5312	12443900	11155100736	13832449811
26B	5382	12773820	11601314350	14385906207
27B	5450	13099528	12047546214	14939348567
28B	5516	13421136	12493762218	15492810889
29B	5581	13739108	12939981218	16046273852
30B	5645	14053525	13386220456	16599720133
31B	5707	14364339	13832441607	17153188106
32B	5767	14671671	14278651078	17706671243
33B	5827	14976174	14724888917	18260128841
34B	5885	15277361	15171100697	18813615816
35B	5942	15575632	15617322681	19367095504
36B	5999	15871123	16063531674	19920590963
37B	6054	16164040	16509743411	20474086254
38B	6108	16454234	16955977213	21027562204
39B	6161	16741875	17402189913	21581061810
40B	6213	17027016	17848425479	22134541051

Using a similar argument, we can prove that any positive integer less than 1.09×10^{23} can be expressed as a sum of at most six tetrahedral numbers, and so on.

With these results and asymptotic analysis to be discussed in §3, we attempt to give the following

Conjecture 1. *Any integer, greater than* 343, 867, *is expressible as the sum of at most four tetrahedral numbers.*

3. Asymptotic form

In this section we want to see how $N_k(n)$ behaves as $n \to \infty$, *i.e.*, the asymptotic partitioning of integer n.

Define the density of k-numbers by

$$\rho_k(n) = N_k(n)/n.$$

Figure 1 shows $\rho_3(n)$ and $\rho_4(n)$ as a function of integer $n \leq 40\text{B}$, respectively.

For $k = 1$, $N_1(n) = m - 1$ where m is the largest number with

$$n \geq (m-1)m(m+1)/6.$$

For $k = 2$, we found that $N_2(n)$, with 400 data points uniformly distributed in $0 < n \leq 40\text{B}$, can be fitted to a quadratic form

$$N_2(n) \approx \lfloor 1.457936 n^{2/3} - 10.388235 n^{1/3} + 1169 \rfloor.$$

The error in this fit is ± 64. The first coefficient is understood. From a more elaborate analysis [1] we found the first coefficient ought to be 1.458326. The earlier paper [1] reported a coefficient of 1.457195 with a relative error of about 0.08%. The new result has only a relative error of 0.03%. In fact, this number can be obtained exactly by the method of Hooley [3].

Similar fits to $N_3(n)$ and $N_4(n)$ give leading terms 0.446244 and 0.553752 respectively. We thus make the following

Conjecture 2. *There exist positive constants* c_2, c_3, *and* c_4 *with* $c_3 + c_4 = 1$ *such that as* $n \to \infty$,

$$\begin{aligned} N_2(n) &\sim c_2 n^{2/3}, \\ N_3(n) &\sim c_3 n, \\ N_4(n) &\sim c_4 n. \end{aligned}$$

Our numerical experiments suggest that $c_2 \approx 1.458326$, $c_3 \approx 0.446244$, $c_4 = 0.553752$. Obviously, $c_3 + c_4 \approx 1$, which means $N_3(n)$ and $N_4(n)$ constitute almost all of the tetrahedral partitions up to n. As mentioned above, $N_2(n) \sim c_2 n^{2/3}$ as $n \to \infty$ can be shown using the method of [3].

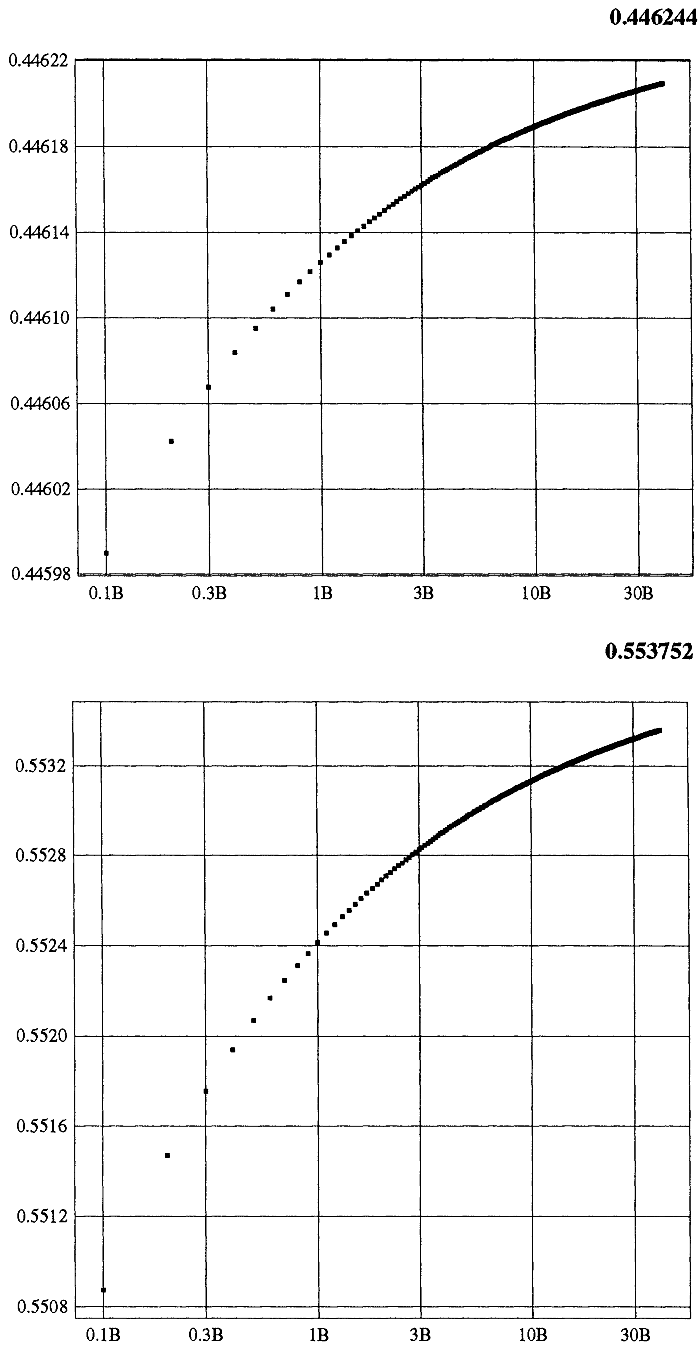

FIGURE 1. These two figures show $\rho_3(n)$ vs. n (top) and $\rho_4(n)$ vs. n (bottom). Asymptotic values ($n \to \infty$) for ρ_3 and ρ_4 are 0.446244 and 0.553752 respectively

4. The search algorithms

We performed all of our decompositions on a 56-node distributed-memory MIMD supercomputer—Intel Paragon XP/S with a local memory of 32 megabytes per node (of which 25 megabytes is user-accessible). Due to the constraint of machine word length, all positive integers greater than two billion are represented by double floating points.

4.1. Sequential algorithm.

This algorithm, naturally revised from [1], only works for n less than $n_{\max} = 3.3$B due to memory limitation.

Before searching, we construct three tables: a 1-number table $J1[j1]$ that contains the sorted lists of all 1-numbers less than $n_{\max}$ (obviously, $j1$ is the running counter for the J1-table), a 2-number table $J2[j2]$, and a 5-number table $J5[j5c]$. For the J5-table, we only tabulate the 241 5-numbers found in [1].

Algorithm 1

(1) Set $j1 = j2 = 1$.
(2) If $n = J1[j1]$, n is a 1-number and set $j1 \Leftarrow j1 + 1$.
Otherwise, next step:
(3) If $n = J2[j2]$, n is a 2-number and set $j2 \Leftarrow j2 + 1$.
Otherwise, next step:
(4) For all $p_1 = J1[j1] < n$, if $n - p_1$ is a 2-number by binary search, n is 3-number.
Otherwise, next step:
(5) If $n - 1$ is a 3-number, but n is not, then n must be a 4-number.
Otherwise, next step:
(6) For all $p_2 = J2[j2] < n$, if $n - p_2$ is a 2-number by binary search, n is 4-number.
Otherwise, next step:
(7) Search n from the 5-number table. If no match, there are two possibilities:
(1) the 5-number table is incomplete; or
(2) the Conjecture 1 is false.

The cost for Steps 2 and 3 is $O(1)$. It is comparatively negligible for the cost of searching all 1- and 2-numbers. For Step 4, the cost for searching in p_1 is $O(n^{1/3})$ and that in $n-p_1$ is $O(\log n)$, leading to a cost of $O(n^{1/3}\log n)$ for checking whether n is a 3-number or not. Therefore, the total cost to check all $k \in [1, n]$ is equal to the summation

$$\sum_{k=1}^{n} k^{1/3}\log k < \int_1^n q^{1/3}\log q dq = C(n^{4/3}\log n - \frac{3}{4}n^{4/3}) \approx Cn^{4/3}\log n,$$

where $C = \frac{3}{4\ln 10}$. Similarly, for Step 6, the cost for searching in p_2 is $O(n^{2/3})$ and that in $n - p_2$ is $\log n$, leading to a cost of $O(n^{2/3}\log n)$ for checking whether n is a 4-number or not. Therefore, the total cost to check all $k \in [1, n]$ is less than the summation

$$\rho_4\sum_{k=1}^{n} k^{2/3}\log k < \rho_4\int_1^n q^{2/3}\log q dq = D(n^{5/3}\log n - \frac{3}{5}n^{5/3}) \approx Dn^{5/3}\log n,$$

where $D = \rho_4\frac{3}{5\ln 10}$ and ρ_4 is the density asymptotic value of a 4-number.

4.2. **Parallel algorithm for large numbers.** Large numbers cause three types of problems: number representation, large memory requirements, and extensive search time. We are constructing a set of fast algorithms—including data distribution for saving space and search distribution for saving time—that will overcome these difficulties. The essence of our new algorithms lies in cutting a large number to smaller ones that need decomposition. The search for small numbers will employ some of the algorithms described in §4.1. Parallelization made storing large tables possible.

The fact that the searching time is *a priori* unknown for a given number makes it difficult to balance loads on the processors. The parallel paradigm we are using is similar to the master-slave method except that the master also performs large-load calculation. When given an interval of integers, we first decompose the interval into an s sub-interval ("grain") with equal number of integers in each. One processor (any one) in the system, we call it the "working master", keeps a list of these grains. To start, every one of the p processors (including the master) is given a grain to work on. If a slave processor finishes its grain it will inform the master which marks the grain as "done". At the same time, the master will issue another grain. This process is repeated by every processor until the last grain is processed. This algorithm has three properties that lead to an extremely high parallel efficiency. First, the number of grains can always be made much larger than the number of processors. Therefore, the load imbalance is invisible. Second, the communication cost to fetch a task (getting a grain from the master) is infinitesimal compared to the time needed to process the grain. There the communication costs are ignorable. As usual, the smaller the grain, the bigger the communication due to more frequent requests to the master for grains, but the smaller the load imbalance. On the other hand, the bigger the grain, the smaller the communication due to less frequent requests to the master for grains, but the bigger the load imbalance. Therefore, there is an optimal value (or a range) for the grain size to achieve maximum parallel efficiency—we choose a grain size of 5 million (5M). Third, the master also sends itself a grain while it is coordinating the slave processors for their grains.

Now, we explain the scheme. Before searching we still construct three tables: a 1-number table $J1[j1]$ that contains the sorted lists of all 1-numbers less than $n_{\max}$ (=40B for the present study), and a 5-number table $J5[j5]$. For the J5-table, we only tabulate the 241 5-numbers found in [1], [2]. During searching we construct two additional "dynamic" tables, a 2-number table $K2[k2]$ in a variable interval in every processor, and a rough 3-number table $K3[k3,p]$, allowing to contain 1- and 2-numbers, in a variable interval in each processor, due to having problem with storing the whole tables of 2-numbers and 3-numbers. Table $K2[k2]$ consists of all 2-numbers in the next interval and we set it as $[N, N+1B]$. However table $K3[k3,p]$ consists of a much shorter interval than table $K2[k2]$ and there are several features that need explanation. This table depends on the processor where it resides and the number the table is used to decompose.

In summary,

Algorithm 2

(1) For each 1B, construct $K2$-table in every processor.
Reset index $k2 = 1$.
(2) For each 5M, assign a new grain in each processor.
Construct $K3$-table.
Reset index $k3 = 1$.
Find the smallest $j1$ such that $J1[j1] > n$.
(3) If $n = J1[j1]$, n must be a 1-number and set $j1 \Leftarrow j1 + 1$: While $n = K3[k3]$, $k3 \Leftarrow k3 + 1$.
Otherwise, next step:
(4) If $n = K2[k2]$, n must be a 2-number and set $k2 \Leftarrow k2 + 1$: While $n = K3[k3]$, $k3 \Leftarrow k3 + 1$.
Otherwise, next step:
(5) If $n = K3[k3]$, n must be a 3-number and set $k3 \Leftarrow k3 + 1$.
Otherwise, next step:
(6) If $n - 1$ is a 3-number and n is not a 3-number, then n must be a 4-number.
Otherwise, next step:
(7) For $r = 1, 2, \cdots, 70$, if $n - J1[r]$ is a 3-number in $K3$-table, n must be a 4-number and stop. If all fails[a], n would be a k-number where $k > 4$.

[a] For our search with n up to 40B, we have not seen this case.

Now, we estimate the complexity of this algorithm. It is obvious that the cost of searching all 3-numbers is reduced to $O(n)$. However there exists a big constant for the complexity due to routine search for a 3-number of new grain. The cost of searching a 4-number consists of two parts: cutting and searching. First, the cost incurred during cutting is finite and small. Second, we show the cost in search *per se* is $O(n)$. Suppose the integer n to be searched satisfies

$$T(m) < n < T(m+1).$$

We then define a remainder $\Delta(r) = n - T(r)$. The job is to confirm at limited r that $\Delta(r)$ is a 3-number. Obviously, $r = 1$ is the best scenario—get the decomposition done at the first cut and the remainder is the smallest possible number required decomposition. If $\Delta(r = 1)$ fails to satisfy the conjecture, move to check $\Delta(r = 2)$, then move to check $\Delta(r = 3)$, until $\Delta(r = r_{\max})$ when the conjecture is satisfied. According to the computation, we find that (a) $r_{\max} = 68$, and (b) most of r is less than 30 to complete the search. In summary, the time to search for all 3-numbers up to n is $O(n)$. The time necessary to check whether n is a 4-number when it is not a 3-number is at worst $O(n^{1/3})$. Heuristically, it is nearly independent of n, *i.e.*, $O(1)$. Therefore, the time to decompose the numbers up to n, heuristically, is $O(n)$.

5. Conclusions

We have addressed three related points in this paper. First, we have for the first time decomposed integers up to 40B by tetrahedral numbers and found at most five tetrahedral numbers are necessary for such a decomposition. Second, we have obtained conjectural asymptotic forms for the decomposition. Third, a more efficient and parallel algorithm is derived. In addition, we make the conjecture that any integer greater than 343,867 is expressible as the sum of at most four tetrahedral numbers.

Acknowledgements

We would like to extend our appreciation to Professor C. N. Yang for encouragement and active discussion in many stages of the project. We thank the anonymous referee for his/her valuable comments.

References

[1] Y. Deng and C. N. Yang, *Waring's problem for pyramidal numbers,* Science in China (Series A) **37**(1994) 277–283. MR **95m:**11109

[2] H. E. Salzer and N. Levine, *Table of integers not exceeding* 1000000 *that are not expressible as the sum of four tetrahedral numbers,* Mathematics Tables and Other Aids to Computation, **12** (1958) 141–144. MR **20:**6194

[3] *C. Hooley, On the representations of a number as the sum of two cubes,* Math Z. **82** (1963) 259-266. MR **27:**5742

Department of Mathematics, National ChangHua University of Education, ChangHua 50058, Taiwan

Center for Scientific Computing, State University of New York at Stony Brook, Stony Brook, New York 11794

URL: `http://ams.sunysb.edu/~deng`