Finding $95800^4 + 217519^4 + 414560^4 = 422481^4$ on the Connection Machine

Roger E. Frye

Thinking Machines Corporation,
245 First Street, Cambridge, Massachusetts 02142-1214
frye@think.com (617)876-1111

The smallest counterexample to Euler's generalization of Fermat's Last Theorem is $95800^4 + 217519^4 + 414560^4 = 422481^4$. *I explain how this solution was found by exhaustive data parallel search on several Connection Machine systems.*

Keywords: biquadrates, Connection Machine, data parallel algorithms, Diophantine equations, Euler conjecture.

1 INTRODUCTION

Noam Elkies [4] found two solutions to the equation $A^4 + B^4 + C^4 = D^4$ in relatively prime natural numbers A, B, C, and D. The smaller solution was $2682440^4 + 15365639^4 + 18796760^4 = 20615673^4$, and the larger had 71 digit numbers in each term. He used a method of parameterized equations which has very little control over the size of the solution, so that there was a possibility that it was not minimal.

I contacted Elkies about the nature of the problem, and he explained the number theoretic constraints which might be sufficient to make an exhaustive search for a minimal solution feasible on a supercomputer. I implemented his suggestions in Common LISP [12] extended with Zetalisp® [1]LOOP [13] for a Symbolics LISP Machine® [2] front end with calls to Paris [2], for the data parallel Connection Machine® [3] [7] system.

This program was run on several LISP machine/Connection Machine systems for more than 100 front end machine hours, until it found the minimal solution, $95800^4 + 217519^4 + 414560^4 = 422481^4$.

Since then, the search algorithm has been improved and the search range has been extended to 2000000^4 for the right hand side of the equation. No other solutions have been found.

The next section of this paper gives an outline of the history of the problem. Section 3 presents a naive approach to exhaustive search for a minimal solution and develops the mathematical constraints needed to limit the search even for a supercomputer. Section 4 discusses the architecture of the data parallel Connection Machine system. Section 5 presents the parallel algorithm. Section 6 contains timings, final results and open questions.

[1] Zetalisp is a trademark of Symbolics, Inc.
[2] Symbolics is a trademark of Symbolics, Inc.
[3] Connection Machine is a registered trademark of Thinking Machines Corporation

2 HISTORY OF ATTACKS ON EULER'S CONJECTURE

Pierre Fermat is famous for an unproved conjecture that the equation $A^n + B^n = C^n$ has no solution in the natural numbers A, B, C if n is an integer greater than 2. This conjecture is generally, though erroneously, called Fermat's Last Theorem. The "Theorem" has not been proved for all $n > 2$, but the $n = 4$ case was proved by Fermat himself [3] .

In 1769, Leonhard Euler extended Fermat's conjecture:

> It has seemed to many Geometers that this theorem [Fermat's Last "Theorem"] may be generalized. Just as there do not exist two cubes whose sum or difference is a cube, it is certain that it is impossible to exhibit three biquadrates whose sum is a biquadrate, but that at least four biquadrates are needed if their sum is to be a biquadrate, although no one has been able up to the present to assign four such biquadrates. In the same manner it would seem to be impossible to exhibit four fifth powers whose sum is a fifth power, and similarly for higher powers.

The term *biquadrate* may be unfamiliar to modern readers; here it means the fourth power of an integer.

Richard Guy gives this quotation and discusses the problem under the heading **D1** in his list of unsolved problems in number theory [6]. He displays the first assignment of four biquadrates, $30^4 + 120^4 + 272^4 + 315^4 = 353^4$, found by R. Norrie in 1911, and the first counterexample to the conjecture, $27^5 + 84^5 + 110^5 + 133^5 = 144^5$, found by Lander and Parkin in 1966. He also writes

> It has not been proved that $a^4 + b^4 + c^4 = d^4$ has no solution in integers; there is none with $d \leq 222,000$. In fact even $a^4 + b^4 + c^4 = d^2$ is unknown,

The first assertion was the result of an exhaustive search by Lander and Parkin on a CDC 6600® [4] [9]. The second assertion is false. Elkies and Guy have exchanged correspondence about the simple solution,

$$(l \cdot m)^4 + (l \cdot n)^4 + (m \cdot n)^4 = [(l^2 + m^2) \cdot n^2 - (l \cdot m)^2]^2$$

whenever $l^2 + m^2 = n^2$, which was known even to Diophantus[5].

Noam Elkies [4] disproved the $n = 4$ case of Euler's conjecture during the summer of 1987. He combined extensive algebraic manipulation with MACSYMA® [5] and a few hours numeric search on a VAX® [6] to find two particular solutions and to prove that there are an infinite number of solutions to the equation $A^4 + B^4 + C^4 = D^4$ in relatively prime natural numbers A, B, C, and D. I can give only the briefest summary of Elkies's method here. He parameterizes a transformation of the equation with a pair of integers constrained by certain Lemmas. His two solutions are on an elliptic curve with the simplest allowed parameterization, namely $(8, -5)$ in his notation. No other solutions have yet been found with his method, although he has shown that the minimal solution reported here has the fairly small parameterization $(20, -9)$.

The existence of Elkies' solution was the starting point for my search for the minimal solution. I read about his result in a note to *sci.math* on the *USENET* news. My initial attempt at an exhaustive computer search to verify whether his solution was minimal fell so far short of the goal that I began corresponding with him via electronic mail. He explained some of his method and concurred that there was indeed a possibility that it was not minimal. He also explained the constraints which could be applied to speed up the search.

During the Fall of 1988, I implemented his suggestions in a program for a LISP Machine front end to a Connection Machine system. I ran this program over a period of a few weeks on several systems of different sizes and types, and on January 8, 1988, found the minimal solution.

[4] CDC 6600 is a trademark of Control Data Corporation
[5] MACSYMA is a trademark of Symbolics, Inc.
[6] VAX is a trademark of Digital Equipment Corporation

3 NAIVE APPROACH TO EXHAUSTIVE SEARCH

How would you write a program to perform an exhaustive search for a minimal solution to $A^4+B^4+C^4 = D^4$? Here is a very simple sample program in Common LISP with Zetalisp LOOP:

```
(defun naive-search (U)
  "Find solution to A^4 + B^4 + C^4 = D^4 < U^4."
  (loop for D from 1 below U do
    (loop for C from 1 below D do
      (loop for A from 1 to C
            for N = (- (biquad D) (biquad C) (biquad A))
            when (biquadrate-p N) do
              (return-from naive-search (list D C A)))))))
```

The program iterates over variables D, C, and A. For each triple, it computes $N = D^4 - C^4 - A^4$. When it finds some N which is a biquadrate, it escapes to the outer level with the identification of the winning values. Details of the definitions of the supporting functions are omitted in order to simplify exposition.

The main problem with this simple program is that it is too slow. It calls the innermost function biquadrate-p about $U^3/6$ times. Even if you had a *super*-supercomputer which could perform the inner loop on 1.0e9 numbers (each up to 75 bits long) in 1 second, then it would take more than 145 days of constant running to find the minimal solution.

This problem is too big just to throw it at a supercomputer; it needs a better algorithm. The required techniques are elementary number theory, and they have long been applied to the related problem of assigning four biquadrates equal to a fifth biquadrate [10,9].

3.1 Modularity Constraints

Consider the fourth powers modulo 5:

N	N^4	$N^4 \bmod 5$
0	0	0
1	1	1
2	16	1
3	81	1
4	256	1

Since the last column contains only zeros and ones, the congruence $a^4 + b^4 + c^4 \equiv d^4$ has only four forms modulo 5:

A^4	+	B^4	+	C^4	$\equiv$	D^4
0	+	0	+	0	$\equiv$	0
0	+	0	+	1	$\equiv$	1
0	+	1	+	0	$\equiv$	1
1	+	0	+	0	$\equiv$	1

The first form may be omitted since all terms are multiples of five and thus are not relatively prime. The last three are the same except for an interchange of variables. As a result D and one other term, say C, must not be multiples of 5, while the other two terms must be multiples of 5. Since A and B are both multiples of 5, the difference $D^4 - C^4$ must be a multiple of $5^4 = 625$.

These constraints may now be combined by considering the possible (D, C) pairs modulo 625. D may take only 500 of the possible 625 residue values. For each value of D, C may take only four values,

corresponding to the four complex factors of $D^4 - C^4$. In order for the factor $D - C$ to be congruent to 0 mod 625, C must be congruent to D mod 625. The factor $D + C$ requires $C \equiv -D \bmod 625$. The factor $D + iC$ requires $C \equiv iD \bmod 625 \equiv 182D \bmod 625$, and finally the factor $D - iC$ requires $C \equiv -iD \bmod 625 \equiv 443D \bmod 625$. In the foregoing, I have used the shorthand i and $-i$ for the two residues such that $residue^2 \equiv -1 \bmod 625$.

The modulo 5 constraints reduce the number of (D, C) pairs which must be considered by a factor of $625/500 \cdot 625/4 = 3125/16 \approx 195$.

When you consider the fourth powers modulo 4 and the resulting possible congruences, you have a similar situation. D and one other term must not be even, while the other two terms must be even. There isn't enough freedom of choice to guarantee that C is both not a multiple of 5 and odd. The only improvement on the number of (D, C) pairs is a factor of 2.

Some further improvements can be obtained by considering moduli 9, 13, and 29. Modulus 9 produces only 4 biquadratic residues which can be combined in only 6 congruences of interest. These congruences may may be summarized by the following two rules in which the overline indicates the negative of an assertion:

when $D^4 \equiv 0 \bmod 9$, then $\overline{C^4 \equiv 0 \bmod 9}$

when $\overline{D^4 \equiv 0 \bmod 9}$ then either $C^4 \equiv 0 \bmod 9$

or $C^4 \equiv D^4 \bmod 9$.

The modulo 9 constraints reduce the number of (D, C) pairs which must be considered by the fraction $3/9 \cdot 6/9 + 6/9 \cdot (3/9 + 2/9) = 16/27$, so the reduction factor is $27/16 \approx 1.7$.

Modulus 13 may be treated in a similar fashion. The resulting rule is that each D^4 residue excludes one C^4 residue: 0 excludes 0; 1 excludes 3; 3 excludes 9; and 9 excludes 1. The reduction fraction is $1/13 \cdot 1/13 + 3 \cdot (4/13 \cdot 4/13) = 122/169$, so the reduction factor is $169/122 \approx 1.4$.

Finally modulus 29 produces two rules: D^4 can not have residue 0, and every other D^4 residue excludes two C^4 residues: 1 excludes 24 and 25; 7 excludes 1 and 23; 16 excludes 7 and 23; 20 excludes 7 and 16; 23 excludes 1 and 24; 24 excludes 20 and 25; and 25 excludes 16 and 20. The reduction fraction is $1/29 + 7 \cdot (4/29 \cdot 21/29) = 617/841$, so the reduction factor is $841/617 \approx 1.4$.

3.2 Common Factor Constraint

When D and C have a (necessarily odd) prime factor, P, in common, we would like to eliminate them since the quotient pair must have been treated earlier. However, if the sum $A^4 + B^4$ does not also have P as a common factor then the (D, C) pair can not be eliminated. $A^4 + B^4 \equiv 0 \bmod P$ when $B \equiv \sqrt[4]{-1} \bmod P$, and this condition occurs if and only if P is of the form $8K + 1$. See Davenport's introduction to the theory of numbers [3] for a proof.

The modularity constraints have already eliminated the pairs with 3, 5, 13, and 29 as common factors. The list of common factors which remain to be eliminated begins $7, 11, 19, 23, 31, 37, \ldots$ and extends to the largest prime $P < \sqrt{C}$.

The reduction factor for (D, C) pairs eliminated by the common factor constraint is quite small. Nevertheless, every pair which can be eliminated by a simple test like this is important, since these pairs do not have to be matched against the set of A values.

3.3 Prime Factor Constraints

Consider a prime factor P of $A^4 + B^4$. If the factor occurs a multiple of 4 times, then the factor could be common to A and B. On the other hand, if the factor occurs some other number of times, then

it can not be common to A and B and must be produced by the sum. Which prime factors can be produced in this manner?

The factor 2 is produced whenever A and B are odd. Odd primes must be of the form $8K + 1$ as we saw in the analysis of common factors in the previous section.

We can apply this information about the prime factors of $A^4 + B^4$ to constrain the (D, C) pairs to be considered. Whenever $D^4 - C^4$ has an odd prime factor which is not of the form $8K + 1$ and which occurs other than a multiple of four times, then that pair should be eliminated.

The reduction factor for (D, C) pairs eliminated by the prime factor constraint is rather hard to calculate. Since $D^4 - C^4$ automatically factors into $(D - C)(D + C)(D^2 - C^2)$, and each factor picks up prime factors more or less independently, the product tends to have many more prime factors than a random number of its size. Empirically, the reduction factor is between 25 and 50.

3.4 Constraints on $A^4 + B^4$

In the previous sections, we discovered several ways in which the numbers produced by the form A^4+B^4 could produce constraints on the (D, C) pairs to be considered. Unfortunately, a reverse application of this method is not as effective, since the form $D^4 - C^4$ is not as limited.

The first constraint is to eliminate the common factor of 5 in A and B. This can be done by dividing $D^4 - C^4$ by 625 before attempting to decompose it into the form $(A^4 + B^4)$.

Let N be the number to be decomposed. The second constraint is to limit $A, B < \sqrt[4]{N}$ rather than to limit $A \leq C$ as was done in the **naive-search** program. In addition to reducing the number of candidates, this constraint eliminates negative candidates.

A third constraint might be that at least one of (A, B) must be even. This would reduce the search space for A by a factor of 2. However, the constraint that one of (A, B) must be at least as large as the other is more powerful, and this constraint is not compatible with the parity constraint. If N is the number to be decomposed, then $N > A^4 \geq B^4$, and the minimum possible value for A would be when A =B, so $2A^4 \geq N$. Therefore $A \geq \sqrt[4]{N/2} = \sqrt[4]{1/2} \cdot \sqrt[4]{N} \approx 0.84\sqrt[4]{N}$. The reduction factor is $1/(1 - \sqrt[4]{1/2}) \approx 6.3$.

Further constraints can be generated for particular residue classes of D and C. For example, one could treat odd and even N separately and apply a parity constraint to the even case. These methods have not been employed since the decomposition into $A^4 + B^4$ does not turn out to be a significant bottleneck in the data parallel program until $D > 1000000$.

Now let's put all of these ideas together into a more reasonable program.

3.5 Constrained Search Program

Even if we combine all of the constraints developed above, the search is likely to take longer than the time a machine can be allocated to the task. Therefore the program should perform the search over some segment of the search space. Furthermore the program separates easily into two passes. In the first pass, it applies constraints to sieve out a set of candidates, and in the second pass, it attempts to decompose each of the candidates. If a decomposition is found, the program returns the decomposition information. Here is a sample LISP program which embodies these ideas:

```
(defun constrained-search (L U)
  "Find solution to A^4 + B^4 + C^4 = D^4
  with L <= D < U."
  (decompose-each (constraint-sieve L U)))
```

This first pass function **constraint-sieve** iterates on the D variable over the values between L and U which are prime with respect to 10. For each D, it iterates on the C variable over the values less than D which are in proper modulo 625 correspondence with D. When a (D, C) pair passes both the set of modularity tests and the common factor and prime-factor tests, then the value $N = (D^4 - C^4)/625$ is put into a list. The lists for each choice of D are appended together and returned as the value of the function. Here is the code for the first pass function:

```
(defun constraint-sieve (L U)
  "Apply modular and factor constraints to D and C.
  Return list of candidates which pass tests."
  (loop for D in (prime-10 L U)
        append
          (loop for C in (good-c-for-d D)
                when (and (mod-ok-p D C)
                          (factor-ok-p D C))
                collect
                  (/ (- (biquad D) (biquad C))
                     625))))
```

The second pass function **decompose-each** takes the list of candidates generated by the first pass as its argument. It iterates on the variable N over the candidates in the list. For each candidate, it computes the proper limits on A and iterates between them. When it finds an A such that $N - A^4$ is a biquadrate, it escapes to the outer level and returns some information about the winner. If a winner is encountered, another function would be needed to fully identify it. Here is the code for the second pass function:

```
(defun decompose-each (candidates)
  "Attempt to decompose each of the candidates.
  Return successful decomposition."
  (loop with half-root = (biquad-rt 0.5)
        for N in candidates
        for a-limit = (i-biquad-rt N)
        for a-min = (ceiling (* half-root a-limit))
        do
    (loop for A from a-min below a-limit
          when (biquadrate-p (- N (biquad A)))
             do
                (return-from decompose-each (list N A))))))
```

How many candidates can we expect the first pass to produce?

$$U^2/2 \cdot 16/3125 \cdot 1/2 \cdot 16/27 \cdot 122/169 \cdot 617/847 \cdot 1/25 \approx U^2/63000$$

How many A values does the second pass have to scan for the average candidate? Assume that the candidates are evenly distributed below U^4, then the average upper limit on A is about $4/5 \cdot (U^4)^{1/4} = 4U/5$ and only 16% of that range is scanned.

In summary, this program calls the innermost function biquadrate-p about $U^2/63000 \cdot 4U/5 \cdot .16 \approx U^3/490000$. Now if you had a supercomputer which could perform the inner loop on 1.0e6 numbers in 1 second, then it would take less than 2 days of constant running to find the minimal solution. This sounds possible. It's time to look at the supercomputer.

4 THE CONNECTION MACHINE ARCHITECTURE

The Connection Machine architecture is a data parallel computing system [1]. A Connection Machine system (CM for short) consists of a set of processors and memories divided into up to four sections of 8192 or 16384 processors and connected through a programmable cross bar switch to up to four external front end computers. The CM-1 Model has 4096 bits of memory per processor, and the CM-2 Model has 65536 bits per processor. Memory is addressable at the bit level and memory fields can be any length. In virtual processing mode, the user sees a larger number of processors, each with a correspondingly smaller memory.

The CM-2 Model has an optional floating point accelerator associated with every 32 processors. It also has optional I/O controllers and framebuffers. The CM-2 may optionally be divided into sections of 4096 processors.

When 65536 processors are operating in parallel, each performing a 32-bit integer addition, the CM-1 operates at about 1000 Mips. (This figure includes all overhead for instruction issuing and decoding.) The CM-2 operates at 2500 Mips. When the CM-2 is equipped with the floating point accelerator, it operates at 2500 MFlops for double precision operands.

The fastest operations such as arithmetic, data manipulation, and data broadcast operations are performed in parallel on all or a selected set of processors and take constant time. Operations which combine the values in selected processors, such as enumeration, rank, scans, and global operations, take time logarithmic in the number of processors. Most of these cooperative operations return a value to the front end computer and thus interrupt instruction pipelines. Operations which move data between processors are slightly more expensive. The most efficient of these operations perform communication between neighbors on a virtual N dimensional grid, while the most general communicate data between any processors and handle collisions. The slowest operations are those which operate entirely on the front end computer or which transmit data serially between individual processors and the front end computer.

The user interacts with one or more ganged sections of the CM through a front end computer, which may be a Symbolics 3600 LISP machine, a DEC VAX 8000 series computer with a BI bus, or a SUN 4 workstation® [7]. The front end provides the program development and execution environment. It issues instructions to one or more microsequencers which operate in parallel and broadcast the lowest level instructions to the data processors.

The high-level languages available for programming the Connection Machine system are Fortran, C*, and *Lisp. These languages are based on well-known standards with minimal extensions to support data parallel constructs. The assembly language for the CM is Paris. This is the target language of the high-level language compilers. Paris functions are available in C and in LISP, and Paris commands may be executed in a UNIX® [8] or LISP environment.

5 PARALLEL IMPLEMENTATION

The implementation of the exhaustive data parallel search on the Connection Machine system is similar in concept to the constrained-search program described above in Section 3.5. It is implemented in LISP for the Symbolics Lisp machine front end with calls to Paris for either the CM-1 or the CM-2. The candidates sieved out by the first pass are collected in a list on the front end for the second pass.

[7] Sun workstation is a registered trademark of Sun Microsystems
[8] UNIX is a trademark of AT&T Bell Laboratories

5.1 Parallel Constraint Sieve

The major theme for the first pass is the problem of when to treat the primes in parallel and when to treat the (D, C) pairs in parallel. It isn't possible to extract all of a class of primes from all of the pairs at the same time. The program must iterate either on the primes or on the pairs in order to perform the data parallel operation on the the other type of data. The best efficiency is obtained by iterating over a small amount of data and performing the parallel operation on a large amount of data. We solve the problem by treating a batch of $(D, C]$ pairs in parallel at first, and then when most of the pairs have been sieved out, treating them individually and the primes in parallel.

The modulo 5 and modulo 4 constraints on (D, C) result in a modularity table of 500 residues (D mod 1250) each paired with 4 residues (C mod 625). These 2000 pairs are written to the CM, one per processor and duplicated as many times as there are processors available. In each successive copy, the C mod 625 residues are incremented by 625. This table facilitates loading of (D, C) pairs. Here is a list of the data processor memory fields involved:

Field Name	Field Usage
d-offset-field	Offsets on D from table
c-offset-field	Offsets on C from table
table-flag	Select all table entries
ok-field-flag	Select (D, C) candidates ll
d-field	D
c-field	C
d4-field	D^4
c4-field	C^4

How are a batch of (D, C) pairs loaded into the CM? Say 16384 processors are attached to the front end. There are 8 copies of the modularity table loaded into **d-offset-field** and **c-offset-field**, and **table-flag** is set to 1 in the first $8 \cdot 2000 = 16000$ processors. Say that the first pair in the batch is $(400000, 345000)$. The D values are loaded by broadcasting 400000 to **d-field** and adding **d-offset-field** to it. Similarly, the C values are loaded by broadcasting 345000 to **c-field** and adding **c-offset-field** to it. The next batch will start with $(400000, 350000)$. If the C values get beyond the D values in some processors, then those processors are adjusted.

With the (D, C) pairs arrayed in the processors, the other modularity constraints are easy to perform. The **ok-field-flag** is initialized with the **table-flag**. The modulus is broadcast to a temporary field in all of the selected processors. The residues for each D^4 and each C^4 are calculated in two other temporary fields. Then the particular rules for the different cases are selectively applied to the pairs to which they apply. The result is that some of the processors are deselected and the bit for those processors in **ok-field-flag** is reset to 0.

Common factor restriction begins with the (D, C) pairs in parallel. The first 30 or so primes which are not of the form $8K + 1$ except for 3, 5, 13, and 29 are successively used to divide D and C. All remaining processors which have both residues equal to zero are deselected.

Prime factor restriction also begins with the (D, C) pairs in parallel. The first 30 or so primes which are not of the form $8K + 1$ are successively used on a field containing $N = (D^4 - C^4)/625$. First all fourth powers of the prime are divided out of N. The the residue modulo the prime is calculated. All remaining processors in which this residue is non-zero are deselected.

At this point, we switch to treating the pairs serially and the primes in parallel. We use tables of odd primes whic are computed at initialization time by several iterations of a parallel sieve of Eratosthenes interspersed with communication instructions for packing the results. After the primes have been calculated and packed into memory, it is easy to select those which are not of the form $8K + 1$.

The remaining (D, C) pairs are now matched against the prime number tables. The pairs are enumerated and successively extracted from CM memory into the front end. The application of common factor and prime factor restraints is quite similar to previous descriptions. Each (D, C) pair which passes these two remaining tests, causes the corresponding $(D^4 - C^4)/625$ value to be pushed on a list for the second pass.

5.2 Parallel Decomposition

The primary data structure for the second pass is an array of all of the A^4 values which could be used to decompose any of the candidates in the list of $(D^4 - C^4)/625$ values from the first pass. Each column of the array is a memory field extending over all of the processors. Each row is another memory field. Let U be the maximum value in the list, then the upper limit on A values is $\lfloor\sqrt[4]{U}\rfloor$. Let L be the minimum value in the list, then the minimum A value is $\sqrt[4]{\lceil L/2\rceil}$. The A values are not stored; they are only used to calculate the A^4 values which are stored in the array.

The program iterates over successive values in the list of candidates. For each candidate N it iterates over the array columns which contain A^4 values which could be used to decompose N. For each column it computes $M = N - A^4$. If $M = (\lfloor\sqrt[4]{M}\rfloor)^4$, then M is a wining biquadrate.

The major theme for the second pass is to avoid checking whether a solution has been found for as long as possible. Solutions are very rare, and it takes time to check. The equality condition indicating a winner is stored as a 1 in a flag field for the winning processor(s). This flag is kept as a logical *OR* of the values for all candidates. At the end of the pass, the flags in all of the processors are combined, and if any of them was a 1, a winner is declared.

If a winner is declared, a modified version of the search program is used to pin it down.

6 TIMINGS, FINAL RESULTS AND OPEN QUESTIONS

Here is some sample output from the first pass of the parallel search program. Explanatory *comments* have been inserted. Many diagnostic outputs which are normally turned off have been switched on in order to show the progress at each stage. With the diagnostics turned off, the program runs much faster than the time stamps would indicate. The pass was run on a 16384 processor CM-1 system. The interval being checked starts with $(D, C) = (500000, 495000)$ and has an upper cutoff limit of $D \leq 501250$.

Initialize the CM.
Cold booting 02/19/1988 05:07:36
Warning: If you are using *Lisp, you must now do *COLD-BOOT.
Initialize one prime number table in the CM.
This output is from an early version with only one table.
Filling CM with primes 02/19/1988 05:07:41
Highest prime found = 180511 02/19/1988 05:07:42
Initialize the (D, C) modularity table in the CM.
Filling CM with modularity table 02/19/1988 05:07:42
8 copies of size 2000 of modularity table fit into CM
Fill the CM with the first batch of (D, C) pairs.
Ensure that $C < D$. For this batch, all of them are.
d-c-c<d leaves 16000 ok entries
Apply modularity constraints.
d-c-mod-9 leaves 9362 ok entries
d-c-mod-13 leaves 6588 ok entries
d-c-mod-29 leaves 4621 ok entries
Apply serial common factor constraints.
Common factors leave 4485 ok entries
Apply serial prime factor constraints.
some-primes leave 391 ok entries
Enumerate remaining pairs. Apply parallel factor constraints.
Checking 391 ok entries
Report total candidates in output list so far.
Now have 87 valid n625
Fill the CM with the second batch of (D, C) pairs.

d-c-c<d leaves 13750 ok entries
d-c-mod-9 leaves 8296 ok entries
d-c-mod-13 leaves 5909 ok entries
d-c-mod-29 leaves 4099 ok entries
Common factors leave 3991 ok entries
some-primes leave 303 ok entries
Checking 303 ok entries
Now have 148 valid n625
About to exit, report total number of candidates found.
Now have 148 valid n625 02/19/1988 05:07:59
The function leaves the candidate list in a global variable.
It does not return any value of interest.
NIL

The discovery of the minimal solution originally took over 100 hours of running time. This could have been much shorter. Some of the algorithmic short cuts described here had not yet been implemented. Several batches were run multiple times. Some batches were run at inefficient virtual processor ratios, and some were run on very small machines.

Using outputs from partial runs, I estimate that the current algorithm running on a 16384 processor CM-1 would take about 33 hours.

David Plummer [11] has used this program as a model and implemented the search on a network of SYMBOLICS Lisp Machines. He verified the minimal solution in 3300 LISP machine hours.

There are several improvements that could be made to the program. The current method starts the search with (D, C) pairs. Noam Elkies[5] has suggested a method which starts with the pairs $(D + C, D - C)$. This method could be several times faster. A new virtual processor scheme being designed at Thinking Machines should allow better management of variables and result in a speedup.

It would be nice to know how sparsely separated solutions are. Is there no other solution between the minimal one and Elkies' smaller solution?

Another realm of exploration would be to investigate the higher powers of Euler's twice refuted conjecture.

ACKNOWLEDGMENTS

I wish to thank several people who helped with the search for solutions. Without Noam Elkies I would not have known where to start. He very patiently explained the number theory to me, and when I still couldn't understand, Jill Mesirov and Washington Taylor explained it again. Donna Fritzsche found the announcement of the minimal solution waiting on her terminal one morning after a night's run. Roger Locniskar adapted the program as a system exerciser; whenever he repaired a set of memory boards, he would do a final system check and incidentally extend the range of the search by running this program on them. Carmen Crocker put this manuscript in its final form.

References

[1] *Connection Machine Model CM-2 Technical Summary.* Thinking Machines Corporation. Cambridge, MA. 1987.

[2] *Connection Machine Parallel Instruction Set (PARIS): The LISP Implementation.* Thinking Machines Corporation. Cambridge, MA. 1986.

[3] Davenport, H. *The Higher Arithmetic: An Introduction to the Theory of Numbers.* Dover Publications. New York. 1983.

[4] Elkies, Noam D. "On $A^4 + B^4 + C^4 = D^4$." *Mathematics of Computation*, 1988 (to appear).

[5] Elkies, Noam D. Personal communication. January 1988.

[6] Guy, Richard K. *Unsolved Problems in Number Theory.* Springer-Verlag. New York. 1981.

[7] Hillis, W. Daniel. *The Connection Machine.* MIT Press. Cambridge, MA. 1985.

[8] Hillis, W. Daniel, and Guy L. Steele Jr. "Data Parallel Algorithms." *Communications of the ACM.* Vol. 29, No. 12, 1986, pp. 1170-1183.

[9] Lander, L. J., T. R. Parkin and J. L. Selfridge. "A Survey of Equal Sums of Like Powers" *Mathematics of Computation.* Vol. 21, 1967, pp. 446-459.

[10] Leech, John. "On $A^4 + B^4 + C^4 = D^4$" *Proc. Cambridge Philos. Soc.*. Vol. 54, 1958, pp. 554-555.

[11] Plummer, David C. Personal communication. March 1988.

[12] Steele, Guy L. Jr. *Common LISP: The Language.* Digital Press. Burlington, MA. 1984.

[13] *Symbolics Common Lisp: Language Dictionary.* Symbolics, Inc. Cambridge, MA. 1986.