Results of computer search for a perfect cuboid

Robert D. Matson

Abstract

A suite of optimized computer programs was designed to systematically search for a perfect cuboid, keep track of close misses, and investigate the statistical trend of these near matches with increasing Euler brick size. While no perfect cuboid was found, the minimum length of the odd side has been substantially extended from the prior published limit of 3 trillion[1] (3×10^{12}) to 25 trillion (2.5×10^{13}), and the minimum side has increased from the prior published limit of 10 billion[2] (10^{10}) to 500 billion (5×10^{11}).

Background

The Euler brick is named after Leonhard Euler, and is a rectangular parallelepiped (or cuboid) which has integer dimensions for not only its length, width and height, but also its three face diagonals. The smallest Euler brick was discovered by Paul Halcke in 1719 and has edge dimensions of 44, 117 and 240, and face diagonals of 125, 244 and 267. A *primitive* Euler brick is an Euler brick whose edge lengths are relatively prime. Note that there are an infinite number of primitive Euler bricks, and that all primitive Euler bricks have one odd edge length, and two even edge lengths.

A *perfect cuboid* is an Euler brick whose space diagonal also has integer length. Stated mathematically, if there is a perfect cuboid it must satisfy the following set of Diophantine equations:

$$\boldsymbol{a^2 + b^2 = d^2}$$

$$\boldsymbol{a^2 + c^2 = e^2}$$

$$\boldsymbol{b^2 + c^2 = f^2}$$

$$\boldsymbol{a^2 + b^2 + c^2 = g^2}$$

As of January 2015, no example of a perfect cuboid has been found, but no one has proven that none exists. Past exhaustive computer searches have determined that the smallest edge must be greater than 10^{10} [R. Rathbun[2]], and the odd edge must be greater than 3 x 10^{12} [B. Butler[1]]. The main results from this paper are that no perfect cuboid exists with an odd edge smaller than 2.5 x 10^{13}, and the smallest even side must be greater than 5 x 10^{11}.

Some interesting facts are known about the dimensions of a primitive perfect cuboid (assuming one exists) based on modular arithmetic. As mentioned above, one side must be odd and the other two sides must be even. Additionally:

- Two edges must be divisible by 3, and at least one of those edges must be divisible by 9.
- Two edges must be divisible by 4, and one of those must be divisible by 16.
- One edge is divisible by 5.
- One edge is divisible by 7. [T. Roberts[3]]
- One edge is divisible by 11.
- One edge is divisible by 19. [T. Roberts[3]]

Bill Butler's algorithm took advantage of the divisibility-by-16 requirement; the author's algorithm exploits this feature as needed, but more often uses the more restrictive divisibility-by-19 requirement.

Algorithmic Approach

The core of the algorithmic approach is the same as that pioneered by Butler, but with some enhancements to increase the search speed while running with essentially unlimited precision. (Butler's program carried 64-bit precision, but was designed in such a way to err on the side of generating a false positive rather than skipping over a real solution. Even so, his program did not generate any false positives. Since I wished to keep track of near-misses, my algorithm supports up to 180 bits of precision.)

Butler's approach takes advantage of the fact that all Pythagorean triples <*X*, *Y*, *Z*> (where *X* is the odd side, *Y* is the even side, and *Z* is the hypotenuse) obey the following set of equations (Euclid's Formula):

$$\boldsymbol{X = (P^2 - Q^2) \times K = (P - Q) \times (P + Q) \times K}$$

$$\boldsymbol{Y = 2PQK}$$

$$\boldsymbol{Z = (P^2 + Q^2) \times K}$$

where *P*, *Q* and *K* are integers. Since *X* is odd, *K* must be odd, and either *P* or *Q* must be odd (but not both). Butler's algorithm finds all possible values for the even side, *Y*, based on all possible integer triplet products <*D1*D2*D3*> that equal *X*:

$$\boldsymbol{D_1 = P - Q}$$

$$\boldsymbol{D_2 = P + Q}$$

$$\boldsymbol{D_3 = K}$$

(Note that D_2 is always greater than D_1.) Solving for *P*, *Q* and *Y*:

$$\boldsymbol{P = \left(\frac{D_1 + D_2}{2}\right)}$$

$$\boldsymbol{Q = \left(\frac{D_2 - D_1}{2}\right)}$$

$$\boldsymbol{Y = \frac{(D_1 + D_2) \times (D_2 - D_1) \times D_3}{2} = \frac{\left({D_2}^2 - {D_1}^2\right) \times D_3}{2}}$$

Substituting $X / (D_1 * D_3)$ for D_2, and simplifying:

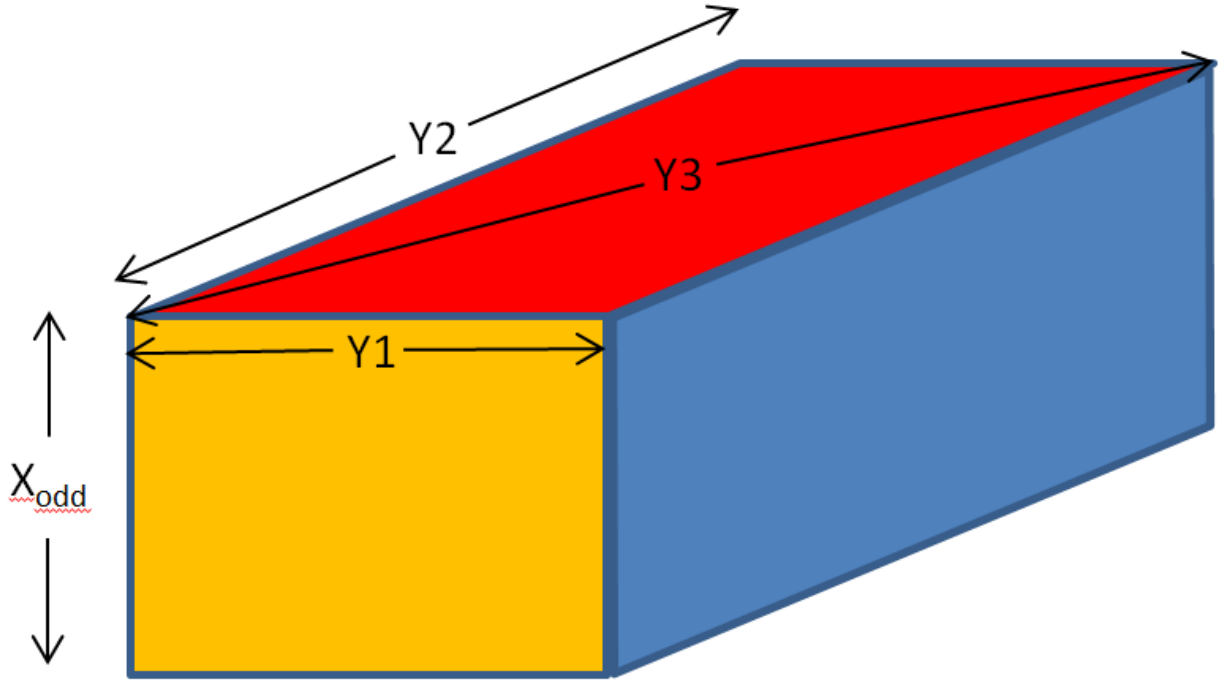
$$\boldsymbol{Y = \frac{X^2 - \left({D_1}^2 \times D_3\right)^2}{2 \times {D_1}^2 \times D_3} = \frac{(X^2 - i^2)}{2i}} \qquad \textbf{(1)}$$

The form of this final expression will be useful later since it indicates how large *Y* can be for a given *X*.

For each *X*, all possible *Y* values are stored in a list. The more prime factors that *X* has, the greater the number of possible *Y* edges. (For *X* values over 4 trillion, the number of *Y* edges can exceed 100,000.) In Butler's algorithm, this *Y* candidate list is further subdivided into those that are divisible by 16 and those that aren't. As discussed earlier, at least one side must come from the divisible-by-16 group, so this subdivision eliminates a lot of potential combinations, speeding up processing. This methodology is also used in the author's program, but only about 5% of the time. If *X* is not divisible by 19 (which happens 94.7% of the time), then one of the sides must be. So in these cases, instead of keeping track of *Y*s that are divisible by 16, the code subdivides the list based on divisibility by 19. (Note that this is much more restrictive than divisibility by 16

since all *Y*s are known to be divisible by 4.) The *Y*s are divided by 4, squared, and stored in a separate $Y^2/16$ list.

The final step is to compute the sums of all possible pairs of values in the $Y^2/16$ lists, where one pair member is always from the divisibility by 19 (or 16) group. The result of this addition is then checked against all other members in the $Y^2/16$ list for a match. The following graphic helps illustrate why this works:



If a perfect cuboid exists, then Y_1 and Y_2 will both be in the *Y* edge list, but so will Y_3. So if the sum of any pair of values in the $Y^2/16$ list can be found elsewhere in that same list, then a perfect cuboid has been found.

Hashing

The *Y* lists can get quite large when the *X* values are the products of many small primes, requiring the number of $Y^2/16$ comparisons to grow in proportion to Y^3. Butler devised a clever system for greatly decreasing the number of required comparisons by building a 14-bit (0-16383) hash table of the $Y^2/16$ values. For each pair of *Y* values, if the sum of their hash values cannot be found in the hash table, then that combination can be skipped. If the hash sum *is* found in the hash table, then it is still only necessary to check those few cases that share the target hash value.

Eliminating the High-*Y* Solutions

Precision and to some extent computational speed are driven by how large the *Y* values can become. Examining equation (1), it is apparent that maximum *Y* occurs when the <D_1, D_2, D_3> triplet product for *X* is <1, *X*, 1>. In this case, *i* = 1, and

$$\boldsymbol{Y_1 = \frac{(X^2 - 1)}{2}} \qquad \textbf{(2)}$$

So if *X* values up to 10^{14} are to be checked, *Y* could be as large as 5 x 10^{27} (93 bits). Since the algorithm must compare the square of one candidate *Y* to the sum of the squares of two other candidate *Y*s, carrying full precision would appear to require 185 bits. However, all candidate *Y*s are divisible by 4, and therefore all candidate Y^2 values are divisible by 16, so the lowest four bits are always zero. So, only 181 bits are needed to test up to 100 trillion. Butler's algorithm uses the C function *hypot*, and flags any case where the computed hypotenuse differs from a target *Y* value by less than one part in 10^{12} (i.e. 40 bits of precision). While Butler's algorithm is not too concerned with how large *Y* and Y^2 can be, the full-precision calculation would benefit if the largest *Y* values could somehow be ruled out as potential solutions. Fortunately, they can.

After <1, *X*, 1>, the next two highest possible solutions for *Y* are based on the triplets <1,*X*/3,3> and <1, *X*/5, 5>. Equation (2) can only be a solution if it is the hypotenuse of two smaller solutions to equation (1). But even the sum of the next two smaller solutions is not as large as (X^2-1) / 2:

$$\boldsymbol{\left(1, \frac{X}{3}, 3\right) \rightarrow Y_3 = \frac{(X^2 - 9)}{6}}$$

$$\boldsymbol{\left(1, \frac{X}{5}, 5\right) \rightarrow Y_5 = \frac{(X^2 - 25)}{10}}$$

$$\boldsymbol{Y_3 + Y_5 = \frac{8X^2 - 120}{30} = \left(\frac{4}{15}\right)X^2 - 40}$$

Since the sum of any other pair of Y values is always less than Y_1, Y_1 cannot be a solution. Similar logic can eliminate Y_3 as a solution:

$$\boldsymbol{Y_7 = \frac{(X^2 - 49)}{14}}$$

$$\begin{aligned} \boldsymbol{{Y_5}^2 + {Y_7}^2} &= \boldsymbol{\frac{(X^2 - 25)^2}{100} + \frac{(X^2 - 49)^2}{196}} \\ &= \boldsymbol{\frac{[49 \times (X^2 - 25)^2 + 25 \times (X^2 - 49)^2]}{4900}} \\ &= \boldsymbol{\frac{(74X^4 - 4900X^2 + 90650)}{4900} = \frac{37}{2450}X^4 - X^2 + 18.5} \end{aligned}$$

$$\boldsymbol{{Y_3}^2 = \frac{1}{36}X^4 - \frac{X^2}{2} + 2.25}$$

Since we know X must be large, only the X^4 term matters; the X^2 and smaller terms can be ignored. Since 1/36 is greater than 37/2450, the sum of the squares of Y_5 and Y_7 will always be less than Y_3. Therefore, Y_3 cannot be a solution.

The same procedure can be used to eliminate Y_5 as a solution. Y_7 is the first case where the sum of the squares of the next two smaller Ys exceeds ${Y_7}^2$:

$$\boldsymbol{{Y_7}^2 \approx \frac{X^4}{196} \approx 0.00510\,X^4}$$

$$\boldsymbol{{Y_9}^2 + {Y_{11}}^2 \approx \frac{X^4}{324} + \frac{X^4}{484} \approx 0.00515\,X^4}$$

So Y_7 cannot be *immediately* ruled out as a potential solution.

A separate program was written to systematically eliminate the highest Y solutions. Per equation (1), all Ys are of the form:

$$\boldsymbol{Y_i = \frac{(X^2 - i^2)}{2i}}$$

For there to be a perfect cuboid solution, there must be a set of three odd integers *A*, *B* and *C* such that:

$$\frac{(\boldsymbol{X^2 - A^2})^2}{\boldsymbol{4A^2}} = \frac{(\boldsymbol{X^2 - B^2})^2}{\boldsymbol{4B^2}} + \frac{(\boldsymbol{X^2 - C^2})^2}{\boldsymbol{4C^2}}$$

or more simply:

$$\frac{(\boldsymbol{X^2 - A^2})^2}{\boldsymbol{A^2}} = \frac{(\boldsymbol{X^2 - B^2})^2}{\boldsymbol{B^2}} + \frac{(\boldsymbol{X^2 - C^2})^2}{\boldsymbol{C^2}}$$

The quadratic solution for X^2 is:

$$\boldsymbol{X^2} = \frac{\boldsymbol{K_2 \pm ABC\sqrt{K_2 - K_1 K_3}}}{\boldsymbol{K_1}} \qquad \textbf{(3)}$$

Where

$$\boldsymbol{K_1 = A^2(B^2 + C^2) - B^2C^2}$$

$$\boldsymbol{K_2 = A^2B^2C^2}$$

$$\boldsymbol{K_3 = B^2 + C^2 - A^2}$$

$$\boldsymbol{A} < B < C$$

For any given *A*, there is a finite set of odd values of *B* and *C* that produce real solutions for X^2, and the vast majority of these solutions will not be integers. If no integer solutions are found, Y_A cannot be the even side of any Pythagorean triple.

Occasionally, an integer X^2 solution will exist for a given value of *A*. The lowest *A* for which this happens is *A*=49, *B*=62, *C*=76:

$$\frac{(\boldsymbol{X^2 - 49^2})^2}{\boldsymbol{2401}} = \frac{(\boldsymbol{X^2 - 62^2})^2}{\boldsymbol{3844}} + \frac{(\boldsymbol{X^2 - 76^2})^2}{\boldsymbol{5776}}$$

$$\boldsymbol{X^2 = 115444 = 2^2 \times 7^2 \times 19 \times 31}$$

However, it is not enough that X^2 be an integer; it must also be a perfect square, which 115444 is not. So Y_{49} is not a possible solution.

All values of *A* were tested up through 5000, and none were found to produce integer solutions for *X*. Table 1 below lists all the primitive *A*-*B*-*C* combinations that produced integer solutions for X^2:

A	***B***	***C***	X^2	***X***
49	*62*	*76*	*115444*	*339.7705*
188	189	1398	8278956	2877.3175
205	206	1420	1199332	1095.1402
241	303	396	38556144	6209.3594
256	273	616	2306304	1518.6520
329	399	423	618849	786.6696
422	423	2538	10978119	3313.3245
473	507	1287	81295929	9016.4255
480	481	600	384800	620.3225
496	620	775	9535600	3087.9767
513	532	1074	2571156	1603.4825
688	756	903	1690416	1300.1600
844	846	8651	43912476	6626.6489
988	1188	1209	1224531	1106.5853
1265	1472	1860	13552704	3681.3997
1440	1443	1924	3463200	1860.9675
1485	1764	1925	11642400	3412.0961
1562	1578	2893	11259819	3355.5654
1634	1881	2658	35831169	5985.9142
1776	1936	3531	59400836	7707.1938
1881	1889	17193	1851221889	43025.8282
1911	2418	2964	175590324	13251.0499
1953	2337	3317	162788409	12758.8561
2022	2109	5402	121061521	11002.7961
2054	2314	3713	81622879	9034.5381
2165	2167	10835	96646033	9830.8714
2185	2189	4577	23914825	4890.2786
2185	2189	5225	23914825	4890.2786
2358	3093	3622	2201359239	46918.6449
2509	2702	5356	137483164	11725.3215
2537	3127	3268	40876144	6393.4454
2772	2775	2860	8333325	2886.7499
3075	3090	12669	269849700	16427.1026
3248	3668	3799	28507696	5339.2599
3385	4303	4615	15910177	3988.7563
3584	3997	4676	48822784	6987.3302
3689	3717	16881	648382329	25463.3527
3696	3700	3885	14814800	3848.9999

A	B	C	X^2	X
3696	3885	4004	14814800	3848.9999
3841	4076	11408	21354669424	146132.3695
3952	4123	4836	19592496	4426.3412
3952	4503	7676	851391216	29178.6089
4288	4292	5249	28716736	5358.7999
4617	4647	36727	13824299889	117576.783
4788	4940	6375	33067125	5750.4022

Table 1. All integer solutions for X^2 for A <= 5000.

Note the two very close misses for A=3696 where X^2 = 14814800: X misses being a perfect square by the minimum possible amount (1). 3849^2 is 14814801. The failure to find a solution to equation (1) where $D_1{}^2 * D_3$ is less than 5000 means that the largest that Y can be is:

$$\boldsymbol{Y = \frac{(X^2 - 5000^2)}{10000}}$$

If values of X are to be tested up to 10^{14}, Y can be as large as 10^{24}, and $Y^2/16$ can be as large as 6.25×10^{46} (156 bits). This is a significant decrease from the 181 bits required if the highest Y values had not been mathematically eliminated.

Results

The main program ran on several cores of a Dell XPS 8500 3.40-GHz 64-bit machine for approximately 3 years, with no perfect cuboid solutions found for the odd side, X, less than 9 trillion. In mid-2014, a modified version of the program was compiled to run on the Isaac Newton High Performance Computing (HPC) Cluster at Central Queensland University. This cluster has 31 compute nodes and 544 cores with a theoretical maximum performance of 12.35 teraflops, resulting in a vast increase in the rate that odd-side X's could be checked. My grateful acknowledgement to Tim Roberts for submitting all the odd-side testing jobs to the HPC cluster! By mid-December 2014, the cluster had completed testing of all odd X's from 9 to 25 trillion, with no perfect cuboids found.

A second program ran for several months in 2014 tallying "close matches" in order to investigate trends with increasing X. By far the closest match found (in terms of percentage of matching bits) was for one of the smallest odd sides, X = 25025:

$Edge_1$: 71820 **$Edge_2$: 5088** **Diagonal: 72000**

$$Edge_1^{\,2} + Edge_2^{\,2} = 5184000144$$

$$Diagonal^2 = 5184000000$$

So the mismatch is only 144 out of 5184000000, or 0.0000028%. How close is the space diagonal to being an integer?

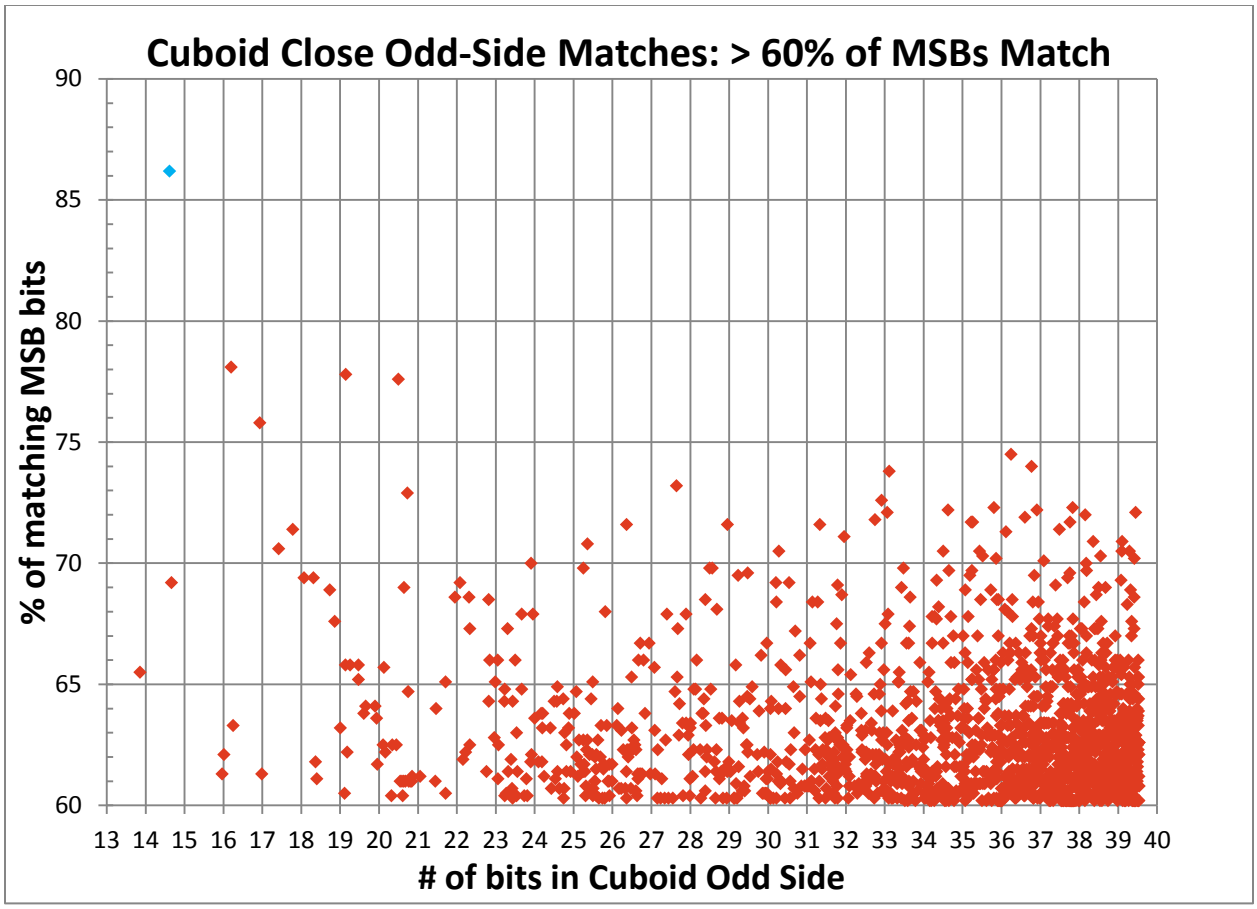
$$\sqrt{25025^2 + 71820^2 + 5088^2} \approx 76225.00094$$

The close-match program keeps track of the number of high-order consecutive bits that match between the sum of the squares of the two sides divided by 16, and the square of the diagonal divided by 16 (remembering that the low order 4 bits will always be zero). For instance, in the above case:

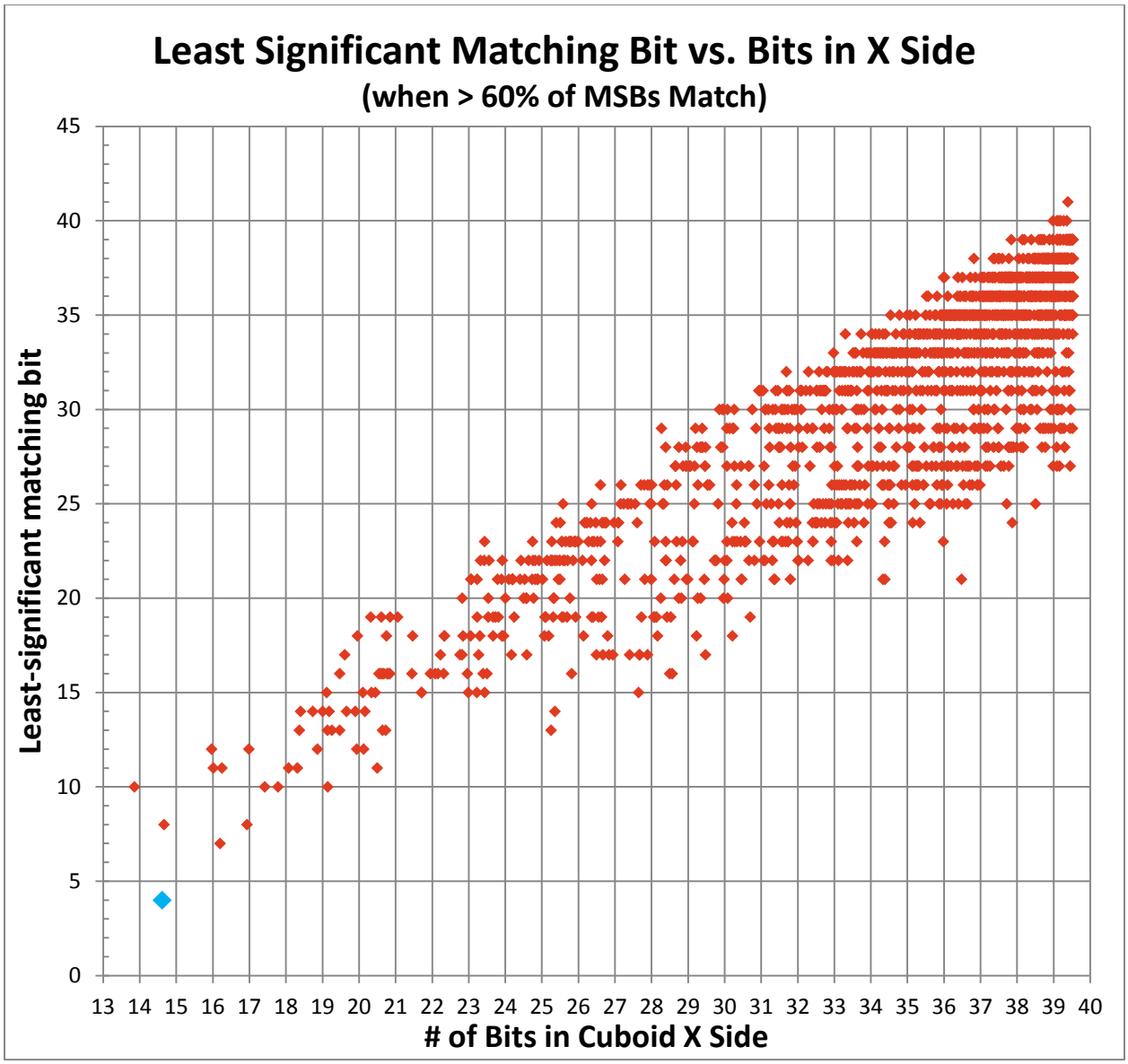
$(Edge_1^{\,2} + Edge_2^{\,2}) / 16 = 324000009 =$ 100110100111111011001000 01001

$Diagonal^2 / 16 = 324000000 =$ 100110100111111011001000 0000

The highest 25 bits match out of a total of 29, or 86.2%. I arbitrarily chose to report any case where greater than 60% of the highest-order consecutive bits matched. A plot of the results for all odd side *X* cases up to 800 billion is shown below:



The blue diamond corresponds to the X=25025 case – the only instance of greater than 80% of bits matching. A perhaps more telling trend is the plot of least-significant matching bit (where all higher bits match). As X increases, the point of Y^2 mismatch (reading from most-significant to least-significant bit) is rising at nearly the same rate as the number of bits in X. This trend suggests that it is highly unlikely that there is a perfect cuboid:



One case worth mentioning is X=117,348,114,345:

Edge1 = 9593 20475 90764 Edge1^2 / 16 = 57 51848 59684 78806 10511 31481

Edge2 = 3 64478 66756 12448 Edge2^2 / 16 = 83027 93694 20127 51527 72432 84544

Diagonal = 3 64604 89393 96864

(Edge1^2 + Edge2^2) / 16 = 83085 45542 79812 30333 82944 16025

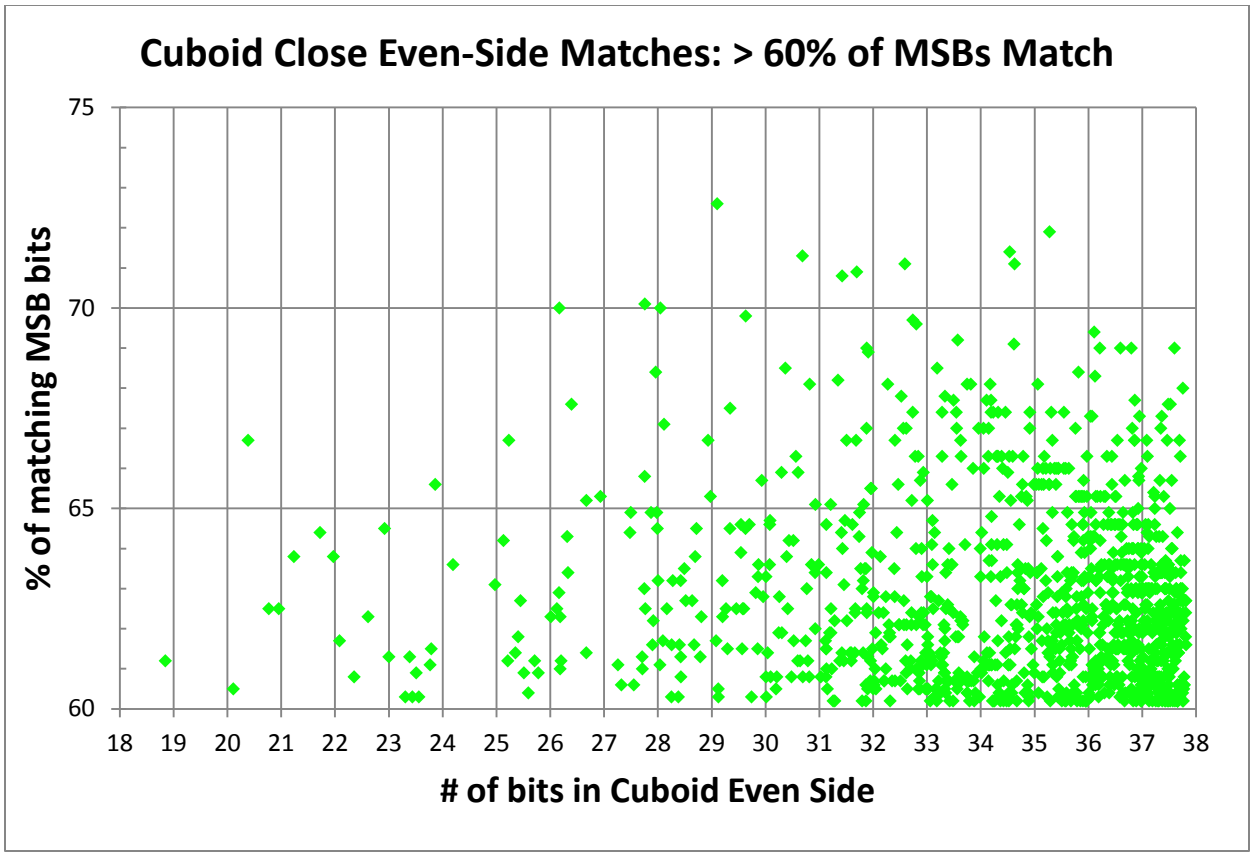
Diagonal^2 / 16 = 83085 45542 79812 30333 82558 14656

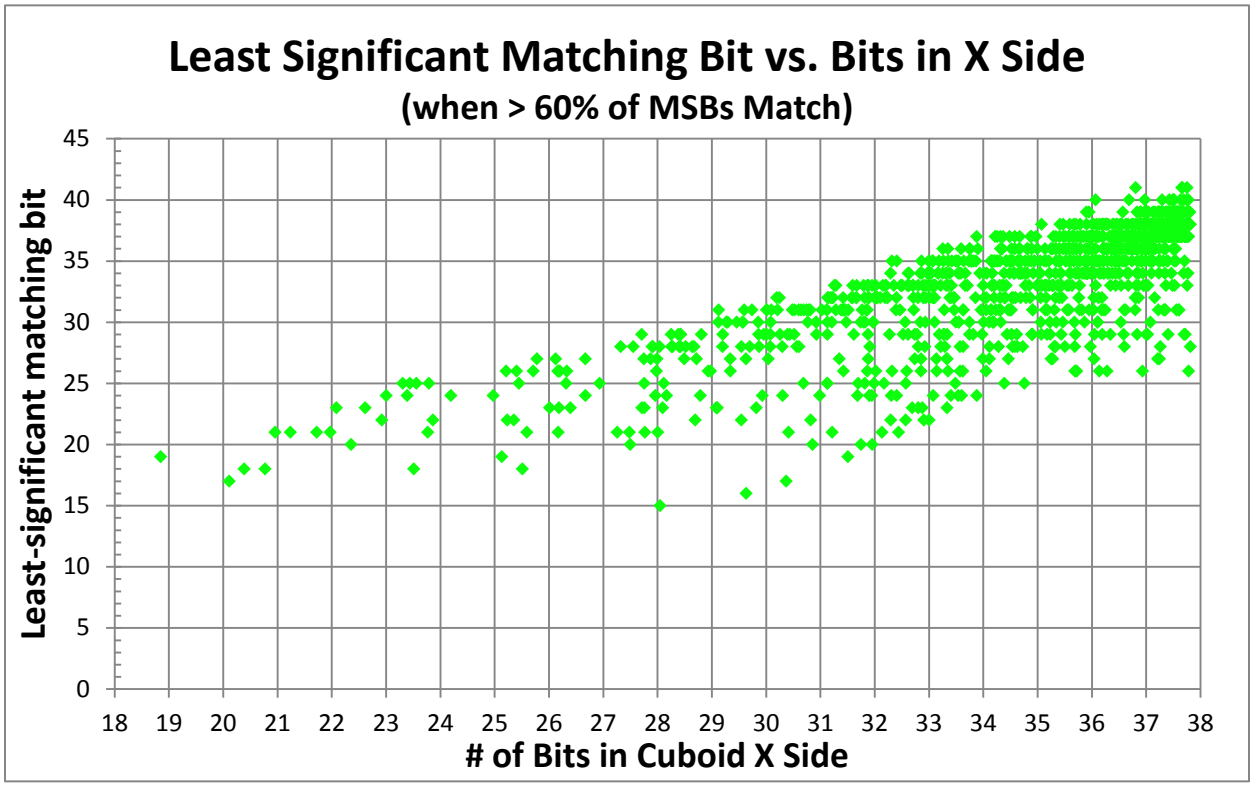
These differ by 38601369, which is less than 1 part in 2 x 10^{22}. The hypotenuse for Edge1 and Edge2 is ≈ 3646048939396864.0000000847. The reason Butler's program did not flag this case is that his hash values are computed from bits 18-31; when the hashes do not match (as is the case here), then the algorithm ignores the case.

Even Side Searches

With the knowledge that there are no perfect cuboids with odd side less than 25 trillion, a separate effort was undertaken to extend the minimum even side (and thus the **overall** minimum cuboid side). Even-side searches are more difficult, primarily because even *X*s will on average have nearly three times the number of divisors of comparable odd *X*s. And since the number of candidate *Y* sides goes up with roughly the square of the number of *X* divisors, it is not surprising that even-side searches are much slower. At least with even-side searches, only those divisible by 4 need to be checked.

The even-side search program has been running on several cores of the same Dell XPS 8500 machine for less than a year, with no perfect cuboid solutions found for the minimum even side up to 500 billion. As with the odd-side searches, another program has been tallying close matches. The highest percentage of consecutive high-order matching bits found so far has been 72.6% (61 out of 84) – quite a bit lower than the 86.2% matching case found for the odd side. A plot of the results for the even side *X* cases up to 200 billion is shown below, followed by the corresponding plot of least-significant matching bit (where all higher bits match). As with the odd side plot, the point of Y^2 mismatch is rising at about the same rate as the number of bits in *X*, again suggesting poor odds that there is a perfect cuboid:


Cuboid Close Even-Side Matches: > 60% of MSBs Match
% of matching MSB bits
75
70
65
60
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
# of bits in Cuboid Even Side


Least Significant Matching Bit vs. Bits in X Side
(when > 60% of MSBs Match)
Least-significant matching bit
45
40
35
30
25
20
15
10
5
0
18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
# of Bits in Cuboid X Side

Summary

There are no perfect cuboids with odd side less than 25 trillion, and no perfect cuboids with minimum side less than 500 billion. While some interesting near-misses have been identified, the overall trend with increasing minimum side does not favor the existence of a perfect cuboid.

References

[1] Butler, Bill (2011). Link: http://www.durangobill.com/IntegerBrick.html.

[2] Rathbun, Randall. “Perfect Cuboid search to 1e10 completed – none found”. NMBRTHRY maillist, November 28, 2010.

[3] Roberts, Tim (2010). “Some constraints on the existence of a perfect cuboid”. *Australian Mathematical Society Gazette* **37**: 29-31.