Experiments in Finding 42 as Sum of 3 Cubes

Martin Gate

May 13, 2019

1 Introduction

The sum of 3 cubes of signed integers has some interesting properties as is summarised in the Wikipedia titled *sum of 3 cubes* in particular for numbers k up to 100 with $k \neq \pm 4 \mod 9$, after finding the solution for 33 given in [1], there is only 42 as the outstanding number that is not known to be the sum of three cubes. This article gives some thoughts on the constraints for the $k = 42$ case and how to encode into a general positive integer. Let

$$x_0^3 + x_1^3 + x_2^3 = 42 \tag{1.1}$$

in the equation we are trying to solve for integers x_0, x_1, x_2.

2 Interesting modular arithmetic results

Taking the mod of a cube provides some interesting results:

Table 2.1: Modular Arithmatic Results

	x									
	1	2	3	4	5	6	7	8	9	42
$x^3 \mod 7$	1	1	-1	1	-1	-1	0			0
$x^3 \mod 6$	1	2	3	4	5	0				0
$x^3 \mod 9$	1	-1	0	1	-1	0	1	-1	0	-3
$x^3 \mod 3$	1	-1	0	1	-1	0	1	-1	0	0

from the $\mod 7$ table one can see that one of the integer powers say x_3 is divisible by 7 so put $x_0 = 7a$. Also note from the $\mod 9$ table that we have, since $42 \mod 9 \equiv -3$

$$X_i^3 \mod 9 \equiv -1 \quad \text{for } i = 0, 1, 2 \tag{2.1}$$

which in turn gives

$$X_i^3 \mod 3 \equiv -1 \quad \text{for } i = 0, 1, 2 \tag{2.2}$$

so we get for $i = 0, 1, 2$

$$x_i \mod 6 \equiv x_i^3 \tag{2.3}$$
$$\equiv 2 \text{or} 5 \tag{2.4}$$
$$\equiv 2 \text{or} -1 \tag{2.5}$$

since $42 \mod 6 \equiv 0$ we get

$$x_0 + x_1 + x_2 \mod 6 \equiv x_0^3 + x_1^3 + x_2^3 \tag{2.6}$$
$$\equiv 0 \tag{2.7}$$

also note that

$$x_0 \mod 6 \equiv 7a \tag{2.8}$$
$$\equiv (7 \mod 6)\, a \tag{2.9}$$
$$\equiv a \tag{2.10}$$

We also get from the $\mod 7$ table since $x_0 \equiv 0 \mod 7$ that

$$x_1^3 \mod 7 \equiv -x_2^3 \tag{2.11}$$
$$\equiv \pm 1 \tag{2.12}$$

so without loss of generality we put

$$x_1^3 \mod 7 \equiv 1 \tag{2.13}$$
$$x_2^3 \mod 7 \equiv -1 \tag{2.14}$$

3 Forming Candidates from a big integer

When working with a Set Partical Swarm Optimiser (SPSO) the cost function is presented with a random big integer that represents in coded form the candidate to cost after it has been modified to meet given constraints.I believe The adoption of the modular constraints avoids attracting the SPSO towards low scoring solutions away from the solution we are here interested in. This section describes how this is done for Equation 1.1.

3.1 partitioning the big integer

For SPSO the parameters big integer is a positive one and can be regarded as an array of bits with least significant bits to the left of the array in our representation; using this the big integer is partitioned as[1]

$$b = j_0 | j_1 | j_2 | f \tag{3.1}$$

where $j_i i = 0, 1, 2$ are positive integers occupying the same number of bits say N with possible padding with zeros on the right as represented. f is regarded as an array of flags taking on the value 1 or 0.

[1]In coding this up the positive big integer is represented as an array of 64 bit words; for the sake of computational speed the partition is applied at word boundaries.

From this we get three signed integers

$$k_i = -1^{f[i]} j_i \quad \text{for} i = 0, 1, 2 \tag{3.2}$$

that provide a starting point in representing the candidate integers for 1.1 by modifying the k_i

$$a = 6k_0 + c_0 \tag{3.3}$$
$$x_1 = 6k_1 + c_1 \tag{3.4}$$
$$x_2 = 6k_2 + c_2 \tag{3.5}$$

where the c_i are yet to be chosen based on f to meet the $\mod 6$ constraints.

Once the k_i are found the the j_i are replaced by $|k_i|$ and the $f[i]$ replaced by $\text{sign}(k_i)$ if required to give the modified big integer that gives the constraint satisfying parameters to be used in the next iteration in the SPSO.

3.2 Choosing the c_i to satisfy the $\mod 6$ Constraints

From 2.10and 2.5 we have c_0 is either 2 or -1 so use $f[3]$ to choose the option and put

$$c_0 = \begin{cases} 2 & \text{if} \quad f[3] = 1 \\ -1 & \text{otherwise} \end{cases} \tag{3.6}$$

the x_1, x_2 must satisfy 2.5 as well as 2.7. It transpires by inspection that x_1 can take on either $\mod 6$ option which in turn determines the x_2 options; to this extent put

$$c_1 = \begin{cases} 2 & \text{if} \quad f[4] = 1 \\ -1 & \text{otherwise} \end{cases} \tag{3.7}$$

and then we have that c_2 satisfies the constraint 2.7 if we use the table 3.1.

Table 3.1: c_2 Values

$f[3]$	$f[4]$	c_2
1	1	2
1	0	-1
0	1	-1
0	0	2

3.3 Changing the k_i to Satisfy the $\mod 7$ Constraints

Finally to satisfy the $\mod 7$ constraints in 2.13 and 2.14 we shift the $\mod 7$ values by adding or subtracting to k_1 and k_2; this does not change the constants c_i and it moves the $\mod 7$ results in reverse order. Since

$$6(x + d) + m \mod 7 \equiv (6x + m) + 6d \tag{3.8}$$
$$\equiv (6x + m) - 1d \tag{3.9}$$
$$\equiv (6x + m) - d \tag{3.10}$$

Let the substitutions be

$$k_1 \leftarrow k_1 + d_1 \tag{3.11}$$
$$k_2 \leftarrow k_2 + d_2 \tag{3.12}$$

then by inspection of the mod 7 table we can satisfy the constraints 2.13 and 2.14 using table 3.2

Table 3.2: d_1, d_2 Values to satisfy 2.13 and 2.14

x_1 mod 7	0	1	2	3	4	5	6
d_1	-1	0	0	1	0	1	2
x_2 mod 7	0	1	2	3	4	5	6
d_2	1	-2	-1	0	-1	0	0

References

[1] Andrew R. Booker. Cracking the problem with 33. *https://arxiv.org/abs/1903.04284*, March 2019.