USING COMPUTER PROGRAMMING AS A DOOR INTO PURE MATH RESEARCH

Jeremiah Bass and Preston Ward
Department of Mathematics, Tarleton State University
P.O. Box T-0470, Stephenville, TX 76402
jbass@tarleton.edu; preston.ward@go.tarleton.edu

1 Introduction

The goal of this paper is to illustrate one way students at the undergraduate level can be provided with a doorway into pure math research. We believe that it is necessary for mathematics students in particular to understand the value of pure math as such, and therefore important to pursue research on that level. Andrew Wiles explains why pure math research is so important:

> ...I don't believe you can have a happily functioning applied maths world without the pure maths to back it up, providing the future and keeping them on the straight and narrow. So it would be very foolish not to invest in pure mathematics. It is a bit like only investing in the energy resources that you can see now. You have to invest in the future . . . . You don't just use up what is there and then start worrying about it when it is gone. It is the same with mathematics; you can't just use up the pure mathematics we have now and then start worrying about it when you need a pure result to generate your applications. [3]

For our purposes, there are a few requirements for getting started. First, we need a problem that is easy enough to state and understand. There are plenty of problems out there, but we need a problem that most undergraduate students should be able to start working on immediately, in order to provide the maximum amount of time for the student to work on the problem before graduating or moving on to a job or graduate school. Moreover, we don't want problems that require a semester or more of background study to even start. Instead, we would like a problem on which something can be done without a lot of initial background theoretical development. Finally, we also want a problem that is "meaty" enough that it will provide the student many opportunities for observation, thought, and reflection, rather than a problem which can be easily solved in a short amount of time.

Though the background of one of the authors is in real analysis and the other is in computer science, we have found that there are many problems from number theory that fit these requirements. Many problems in this area are easy to state and understand. But what is even better is that it is also a fact that for many problems in this area, students can immediately begin to find patterns and solutions with the use of computers. And that is where the technology component of this project comes in.

2 The Technological Component

Technology has been used often in pure math research. However, our approach may be unique in that we have found a way to harness the power of modern graphics processing units (GPUs) to essentially turn a desk-top computer into a super-computer, making it easier to search the infinite spaces we are working in (though not exhaustively, of course).

It is common knowledge that one of the biggest differences between the present and just ten years before is the advancement of home computing power in general. Even without factoring in parallel processing, our standard desktop CPUs are much more powerful than they were just a few years ago. However, being able to cheaply purchase GPUs does effectively turn our desktop PCs into machines that are more powerful than super computers even a decade ago.

What is the big difference between the CPU and the GPU? The CPU has standard between 2 and 8 cores, and each of those cores is capable at doing just about everything. This includes calculations, memory fetching, IO, interrupts, and every operation a computer can do. It thus has a large cache and instruction set. The GPU, on the other hand, has thousands of cores (the count depends on the GPU, but one we used for our project, the Nvidia GTX 1080, has 2560 cores), but they are only good at one thing, and that's doing calculations. So they have a smaller cache and a simpler instruction set. The CPU is like hiring just a few people who have PhD's, and can do a lot of different things very well. The GPU, on the other hand, is like hiring an entire fleet of workers, but they are only able to perform one specialized task. However, thanks to cuda, we can get these workers to perform together, and take large problems and have them all work on it at the same time, greatly reducing computation time, and thus giving us super-computing like capabilities.

3 Using Magic Squares of Squares

As an example of how to intersect technology and pure math research, we will be looking at the problem of the existence of 3×3 Magic Squares of Squares. Of course, one potential drawback of problems like this is that, thought they may be easy to state, they can be very difficult to solve. From a pedagogical perspective, however, this can be a good thing. We want students to have the opportunity to, as Wiles puts it, to stumble around in the dark:

> One enters the first room of the mansion and it's dark. Completely dark. One stumbles around bumping into the furniture, but gradually you learn where each piece of furniture is. Finally, after six months or so, you find the light switch, you turn it on, and suddenly it's all illuminated. You can see exactly where you were. Then you move into the next room and spend another six months in the dark. So each of these breakthroughs, while sometimes they're momentary, sometimes over a period of a day or two, they are the culmina-

> tion of, and couldn't exist without, the many months of stumbling around in the dark that precede them. [4]

What is a magic square? It is a square array of distinct positive integers such that each row, column, and diagonal all have a common sum. For example, the following is an example of a 3×3 Magic Square:

$$\begin{bmatrix} 14 & 1 & 12 \\ 7 & 9 & 11 \\ 6 & 17 & 4 \end{bmatrix}$$

For example, you can generate a 3×3 magic square by taking positive integers n, m, and p and constructing the following array:

$$\begin{bmatrix} p+n & p-n-m & p+m \\ p-n+m & p & p+n-m \\ p-m & p+n+m & p-n \end{bmatrix}$$

In the previous example, we let $p = 9$, $n = 5$, and $n = 3$. Note that for any magic 3×3 square, the common sum must be equal to 3 times the center term.

Now it follows that there are many examples of such squares (in fact, an infinite number of them). A more interesting question therefore is whether or not such a magic square of *squares* exists? Consider the following array:

$$\begin{bmatrix} a^2 & b^2 & c^2 \\ d^2 & e^2 & f^2 \\ g^2 & h^2 & i^2 \end{bmatrix}$$

We want to know if a matrix of this sort exists as a magic square. Of course, to make the problem interesting, we still need each of the numbers in the array to be distinct positive integers. Moreover, we assume that all the entries are relatively prime; otherwise, we could divide out the greatest common divisor to produce a new solution with a smaller magic number. With this assumption, it follows that all the entries $a, b, \ldots, i$ are odd.

If it is a magic square, then $a^2 + e^2 + i^2 = 3e^2$, and so $a^2 + i^2 = 2e^2$. Similarly with all the other "center" sums. This turns out to be a key insight on how to search for such squares and to look for patterns. Our first algorithm will involve factoring numbers of the form $2e^2$ into sums of two squares, and our second will involve factoring numbers of the form $3e^2$ into sums of three squares.

From a historical point of view, this is an interesting question because it goes back at least to Euler and has kept reemerging from time to time [1]. In 1876, Eduard Lucas studied the problem, and showed how to construct semi-magic squares of squares (a square is "semi-magic" if all the rows and columns have a common sum, but not the diagonals) [1].

In 1984, LaBar posed the problem in the College Mathematics Journal (Problem 270), and it has been studied to the present day [2]. However, despite the long history of this problem, as far as we know it is still an open problem whether or not such an object exists.

4 Algorithm 1: Generating the Center Sums

We began by taking a number ($2e^2$) that can be factored into a sum of squares in at least four ways in order to generate the entries for the 3×3 matrix. This will at least guarantee that all the center sums are correct. To do this, we need a number that can factor into a sum of two squares in at least two ways. The first number that meets this criteria is 65, which factors into 13 and 5. Using the identity

$$e = (X2 + Y2)(A2 + B2) = (XB \pm YA)^2 + (XA \mp YB)^2$$

we see that 13 and 5 can be expressed as

$$\begin{aligned} 5 &= 1^2 + 2^2 \\ 13 &= 2^2 + 3^2. \end{aligned}$$

This indicates that $X = 1$, $Y = 2$, $A = 2$, and $B = 3$. Therefore, $e = 7^2 + 4^2 = 1^2 + 8^2$. Based on the center sum equations, we know that

$$2e^2 = a^2 + i^2 = c^2 + g^2 = b^2 + h^2 = d^2 + f^2. \tag{1}$$

Thus,

$$\begin{aligned} 2(65)^2 &= 2(1^2 + 8^2)(1^2 + 8^2) = 47^2 + 79^2 \\ &= 2(1^2 + 8^2)(7^2 + 4^2) = 23^2 + 89^2 = 35^2 + 85^2 \\ &= 2(7^2 + 4^2)(7^2 + 4^2) = 13^2 + 91^2. \end{aligned}$$

These terms generate the following array:

$$\begin{bmatrix} 91^2 & 89^2 & 85^2 \\ 79^2 & 65^2 & 47^2 \\ 35^2 & 23^2 & 13^2 \end{bmatrix}$$

This is not a magic square of squares, however, because while the four central sums equal the same magic number, the others do not.

From there, we implemented this algorithm in python. We first search for a number which can be expressed as a sum of squares in at least four distinct ways. According to the identities referred to in (1), we view this number as $2e^2$. The number e^2 is then placed at the center of a matrix and that matrix is populated by the sums of squares. We check the outer sums to see if the full matrix is a magic square of squares, and if not, we try each permutation of the matrix, and then move on to another central number.

Once the basic algorithm was working, we were able to parallelize the code using cuda and high-powered GPUs in order to process the bigger numbers more quickly. Instead of checking one matrix at a time, we were able to check thousands. We ran this for values of e into the billions before the computations began to slow down dramatically.

One thing we noticed is that this algorithm never produced any squares with more than four correct sums. This is significant, because we know that there are squares with five, six, and even seven correct sums. It is interesting to note that in the cases where you have more than four correct sums, one of the diagonals always fails to sum up correctly. Could this lead to an insight on how to prove that a magic sum could not exist?

5 Algorithm 2: Trading Spaces

In the second algorithm, we take numbers of the form $3e^2$ and see if they can be factored into sums of three squares, and use this to populate a square. However, we have also found it useful to think of this in the context of graphs. And so another way we have explored this problem is by taking each row, column, and diagonal of the matrix and think of them as a node in a graph. If two nodes share exactly one element in common, connect them with an edge. This takes the array on the left and transforms it into the graph on the right (see Figure 1).

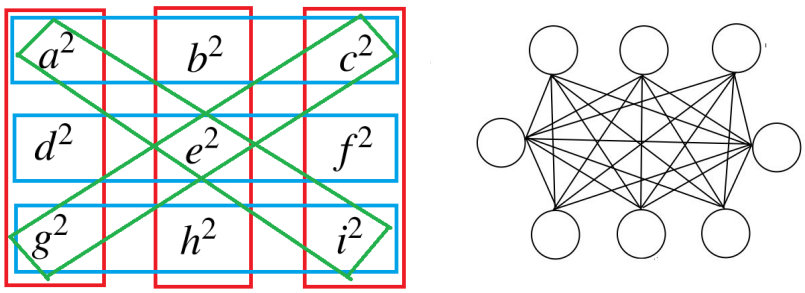

Figure 1: From matrix to graph.

This led us to the following algorithm:

1. Generate a list of sums of three squares for a given number.
2. Once you have them all, make each set of three squares into a node.
3. Iterate over pairs of nodes, and connect them if they have one element in common and the other are distinct.
4. If the final graph has fewer than eight nodes, throw it out.
5. If it has at least eight nodes, start removing nodes that have four or fewer edges, and repeat. If the graph is reduced to zero nodes, throw it out. This is known as core reduction.

6. If the resulting graph has at least eight nodes, check for magic squares of squares.

With this method we have been able to find many squares with five or six common sums, and we believe it will return to us ones with seven common sums. Several squares with seven sums are known, but it is possible that this method of core reduction would make them inaccessible to our algorithm. We intend to look further into this.

In generating these graphs, we have found some interesting patterns. The first class exhibits two clusters of nodes, connected by either one node or one edge (Figure 2):

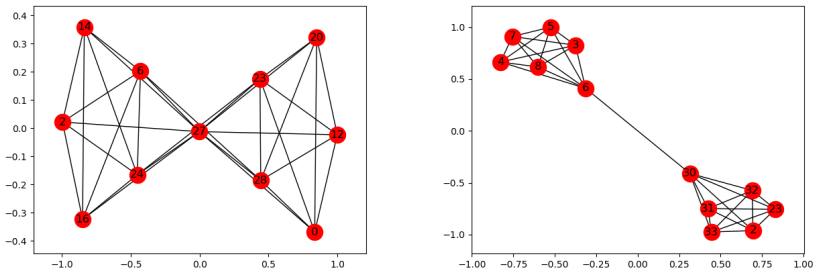

Figure 2: Two clusters, connected by one edge.

In the second class are pairs of disjoint clusters of nodes (Figure 3):

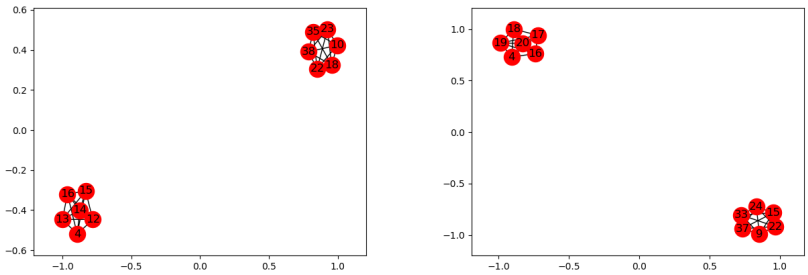

Figure 3: Two disjoint clusters.

In the third class, we find three loosely connected clusters of nodes (Figure 4).

The last class is the most promising, bearing the closest resemblance to our original depiction of a magic square of squares in graph form. However, these graphs have at least one or two extra edges, breaking the property of uniqueness (Figure 5).

6 Concluding Remarks

One of the positive features of problems like this is that it naturally leads to several related problems. For example, one could ask what is the smallest possible magic square of cubes? Or, what is the smallest possible magic cube of

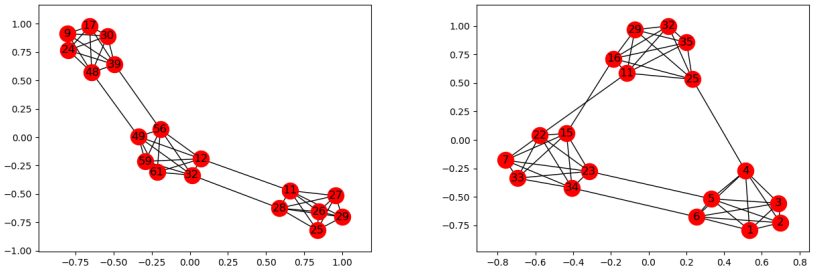

Figure 4: Three loosely connected clusters.

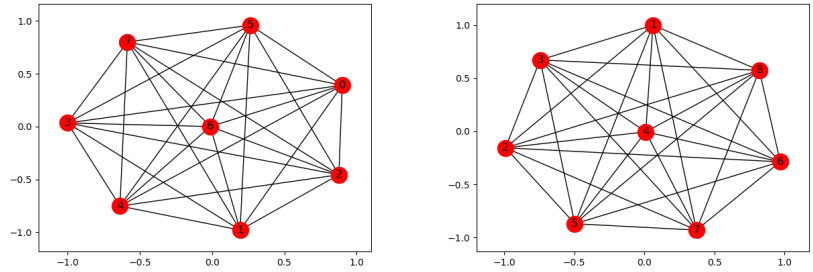

Figure 5: Almost there?

squares? As far as we know, these are also unsolved problems.

From looking at this problem, we think that it is probable that a 3×3 magic square of squares does not exist. That means, importantly, that no matter how much we search the relevant space, we will never be able to prove the non-existence of such an object. There must be a theoretical proof. Hence, this problem not only shows how technology can be used to get a jump-start on problems such as this, but it also demonstrates its limitations. If such a square exists, it is just a matter of finding one. But if none exists, we must be able to show this by writing a proof.

References

[1] Boyer, Christian. "Some notes on the magic squares of squares problem." The Mathematical Intelligencer, vol. 27, no. 2, 2005, pp. 52-64.

[2] Klamkin, M. S., et al. "Problems." The College Mathematics Journal, vol. 15, no. 1, 1984, pp. 68–74.

[3] Raussen, Martin, and Skau, Christian,"Interview with Abel Laureate Sir Andrew J. Wiles," Notices of the American Mathematical Society, vol. 64, no. 3, 2017, p. 206.

[4] Singh, Simon. *Fermat's Enigma*. Anchor Books, New York, 1998.