Experimental Mathematics Techniques for Boolean Functions and Combinatorial Games

Blair Seidler

Rutgers University, New Brunswick

August 14, 2023

What is Experimental Mathematics?

"Mathematics is not a deductive science – that's a cliché. When you try to prove a theorem, you don't just list the hypotheses and then start to reason. What you do is trial and error, experimentation, guesswork. You want to find out what the facts are, and what you do is in that respect similar to what a laboratory technician does, but it is different in its degree of precision and information. Possibly philosophers would look on us mathematicians the same way as we look on the technicians, if they dared." -Paul Halmos

What is Experimental Mathematics?

"Then, about 2300 years ago came a fellow called Euclid, and Euclid ruined mathematics by turning it into a deductive science."

- Doron Zeilberger

What is Experimental Mathematics?

What are the fundamental principles of experimentation which underlie the scientific method? The scientist

- Observes some interesting phenomenon
- Wonders what causes it and starts to reason why it is so
- States a hypothesis
- Designs an experiment with which they obtain evidence
- Notes that the hypothesis is supported or refuted
- Refines and repeats

If all goes well, a satisfactory explanation for the observed phenomenon eventually becomes a first-author paper, a Theory, or possibly even a Law.

What is Experimental Mathematics?

The experimental mathematician follows a similar path. We observe some mathematical phenomenon and wonder what causes it. We program our computers to generate as many cases as possible. Sometimes, this first round of results is enough for us to see what is going on. Other times, we need to rely on the computer to help us analyze its own output. We may need to iterate through several generations of code to achieve the necessary efficiency. But eventually, if all goes well, we have a pattern, an explanation, a conjecture, a theorem...

Overview

Today's presentation includes three projects:

- Minimal Circuits for Boolean Functions of Few Variables
- Statistics on Subcubes of the Discrete *n*-Cube
- Combinatorial Game - Juniper Green

In each case, the objects being studied exhibit exponential growth in some fashion, rendering brute force enumeration powerless at a relatively early stage.

We use the Overlapping Stages technique described by Zeilberger in 2004 to mitigate the exponential growth.

Overlapping Stages

We first write programs which directly encode the definitions of the objects we are studying. These programs are generally quite slow and are only able to compute the smallest cases of the problem, but they have the advantage of being relatively easy to check for accuracy. We then

- Find some pruning technique or symmetry consideration
- Write a (hopefully) faster program
- Check that the smaller cases match those generated by the previous stage
- Repeat until we have enough data or run out of optimizations

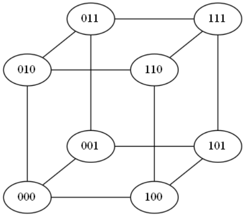
Boolean Functions

Let $K = \{false, true\}$. Then $f : K^n \to K$ is a Boolean function.

In practice, K can be any two-element set. The most common choice is $K = \{0, 1\}$. For reasons which will become clear in a moment, we use $K = \{-1, 1\}$ in our Maple implementation.

Why use $K = \{-1, 1\}$?

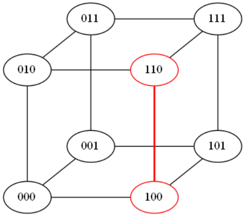
The reason that we use $\{-1, 1\}$ is for ease of referencing subcubes.



This is the standard Hamming cube for K^3

Why use $K = \{-1, 1\}$?

The reason that we use $\{-1, 1\}$ is for ease of referencing subcubes.

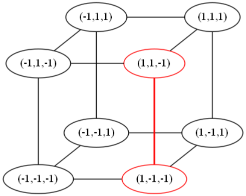


This is the standard Hamming cube for K^3

In much of the literature, the 1-dimensional subcube highlighted in red is written as $(1, *, 0)$, $(1, -, 0)$ or $(1, B, 0)$.

Why use $K = \{-1, 1\}$?

The reason that we use $\{-1, 1\}$ is for ease of referencing subcubes.

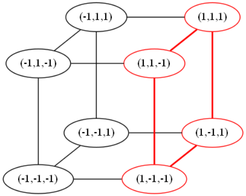


If we instead label this cube with $K = \{-1, 1\}$

The 1-dimensional subcube highlighted in red is written as $(1, 0, -1)$, which we represent in Maple as $[1, 0, -1]$.

Why use $K = \{-1, 1\}$?

The reason that we use $\{-1, 1\}$ is for ease of referencing subcubes.

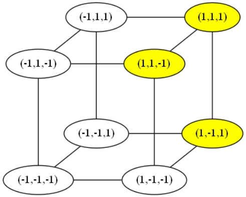


If we instead label this cube with $K = \{-1, 1\}$

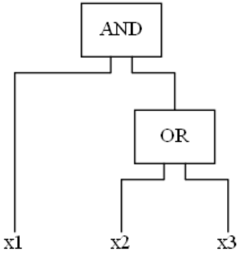
We can write the 2-dimensional subcube highlighted in red as $(1, 0, 0)$, which we represent in Maple as $[1, 0, 0]$.

Representing Boolean Functions

True points in Hamming cube:



Boolean circuit:



Set of true points: $\{[1,-1,1],[1,1,-1],[1,1,1]\}$

DNF: $x_1\overline{x_2}x_3 \vee x_1x_2\overline{x_3} \vee x_1x_2x_3$

Reduced expression: $x_1 \wedge (x_2 \vee x_3)$

Straight-Line Programs - Definition

We use Straight-Line Programs as our model for Boolean circuits. These are programs with no control structures (loops, branches, etc.). Each line is of the form $y_i =< expression >$, and the output of the program is the output of the last line.

We allow a gate to be any of the 10 functions on two variables which depend on both inputs. Other line types handle reading the input and special cases for degenerate functions.

Straight-Line Programs - Gate Types

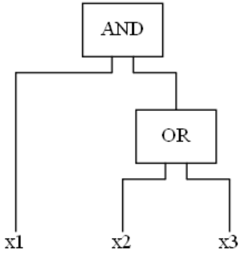
Line format	Function	Context
$[1, i, j]$	Sets $y_k = y_i \wedge y_j$	On line k with $i, j < k$
$[2, i, j]$	Sets $y_k = y_i \wedge \neg y_j$	On line k with $i, j < k$
$[3, i, j]$	Sets $y_k = \neg y_i \wedge y_j$	On line k with $i, j < k$
$[4, i, j]$	Sets $y_k = y_i \oplus y_j$	On line k with $i, j < k$
$[5, i, j]$	Sets $y_k = y_i \vee y_j$	On line k with $i, j < k$
$[6, i, j]$	Sets $y_k = \neg y_i \wedge \neg y_j$	On line k with $i, j < k$
$[7, i, j]$	Sets $y_k = y_i \equiv y_j$	On line k with $i, j < k$
$[8, i, j]$	Sets $y_k = y_i \vee \neg y_j$	On line k with $i, j < k$
$[9, i, j]$	Sets $y_k = \neg y_i \vee y_j$	On line k with $i, j < k$
$[10, i, j]$	Sets $y_k = \neg y_i \vee \neg y_j$	On line k with $i, j < k$

Straight-Line Programs - Gate Types

Line format	Function	Context
$[i]$	Sets $y_i = x_i$	Appears on line i
$[i]$	Outputs y_i $(= x_i)$	After line i (only for 0-gate fns)
$[NOT, i]$	Outputs $\neg y_i$ $(= \neg x_i)$	After line i (only for 0-gate fns)
$[TRUE]$	Outputs 1	Last line of constant function
$[FALSE]$	Outputs -1	Last line of constant function

Straight-Line Programs - Example

So, how would we construct a straight-line program for our earlier circuit example on input $[x_1, x_2, x_3]$?



Straight-line program:

$y_1 = x_1$	
$y_2 = x_2$	Read Inputs
$y_3 = x_3$	
$y_4 = y_2 \vee y_3$	OR gate
$y_5 = y_1 \wedge y_4$	AND gate

In Maple, we write this program as: $[[1], [2], [3], [5, 2, 3], [1, 1, 4]]$

Big Equivalence Classes

We would like to define equivalence classes of functions so we can look for fewer circuits. There are several ways to accomplish this, but we use the scheme from Tilman Piesk's webpage "Equivalence classes of Boolean functions".

A big equivalence class is a set of Boolean functions which are equivalent under some signed permutation of the input variables. (e.g. if $f(x_1, x_2, x_3) = g(\neg x_2, x_3, \neg x_1)$ for all $\mathbf{x} \in K^3$, then f and g are in the same BEC.)

Big Equivalence Classes

We use BEC's to reduce the number of functions we need to consider. The number of Boolean functions on n variables is $\{4, 16, 256, 65536, 4294967296\}$ for $1 \leq n \leq 5$.

The number of BEC's (OEIS sequence A000616) is $\{3, 6, 22, 402, 1228158\}$ for $1 \leq n \leq 5$.

Searching for over a million 5 variable functions is not feasible, but for the 4 variable functions 402 is a much more manageable number than $65,536$.

Big Equivalence Classes

Once we have a circuit for one member of a BEC, it is relatively simple to convert that circuit to accept a different function of the BEC given the signed permutation which maps one to the other. We leave the structure of the gates intact.

If the variables are permuted by $\sigma : [n] \to [n]$, we just switch any gate input $i \in [n]$ to $\sigma(i)$.

If a variable is negated by the signed permutation, we just change the type of any gate to which it is an input (e.g. if an XOR (type 4) gate has one of its inputs negated, change the gate type to EQUIV).

Assigning Integers to Boolean Functions

In order to take advantage of BEC's, we need a way to choose a canonical representative of each class. There are 2^{2^n} functions on n variables, so numbering them from 0 to $2^{2^n} - 1$ seems natural.

We again follow Piesk, using his numbering scheme and choosing the lowest numbered element of each BEC as the canonical representative. These representatives are listed in OEIS sequence A227723. The next slide shows the numbering scheme for 2 variable functions.

Assigning Integers to Boolean Functions

Input	f_0	f_1	f_2	f_3	f_4	f_5	f_6	f_7
$\{-1,-1\}$	-1	-1	-1	-1	-1	-1	-1	-1
$\{-1,1\}$	-1	-1	-1	-1	1	1	1	1
$\{1,-1\}$	-1	-1	1	1	-1	-1	1	1
$\{1,1\}$	-1	1	-1	1	-1	1	-1	1

Input	f_8	f_9	f_{10}	f_{11}	f_{12}	f_{13}	f_{14}	f_{15}
$\{-1,-1\}$	1	1	1	1	1	1	1	1
$\{-1,1\}$	-1	-1	-1	-1	1	1	1	1
$\{1,-1\}$	-1	-1	1	1	-1	-1	1	1
$\{1,1\}$	-1	1	-1	1	-1	1	-1	1

Straight-Line Programs - Number of Circuits

So how big is the haystack? For example, with 4 variables and 7 gates, how many syntactically valid SLP's are there?

For each gate, we have 10 choices for the gate type, and each input can be chosen from every line number smaller than the current one, so we have:

$$\prod_{i=4}^{10} 10i^2 \approx 3.66 \times 10^{18} \text{ valid programs.}$$

Obviously, we will need to reduce the size of the haystack to have any hope at searching through it. We employ several pruning techniques to accomplish this task.

Reducing Number of SLP's

The most obvious optimization is to eliminate any circuit where the output of a gate is unused. Our recursive algorithm for generating cicuits takes care of that by generating the final gate and then creating one (possibly empty) subcircuit for each input.

The next obvious step is to restrict all gates to be of the form $[g, i, j]$ with $i < j$.

If $i = j$, the gate output is either constant, y_i, or $\neg y_i$. Any such gate can be eliminated, meaning that the Boolean function could be computed by a circuit with fewer gates.

If $i > j$, we can reverse the inputs and change the gate type (by swapping types $2 \Leftrightarrow 3$ or $8 \Leftrightarrow 9$) if the gate is not symmetric with respect to its inputs.

Reducing Number of SLP's

A slightly less obvious simplification is to eliminate circuits and subcircuits which are mirror images of each other. If the final gate G (which produces the overall output of the circuit) has two inputs representing subcircuits A and B, then there is another circuit whose final gate has B as the first input and A as its second which is functionally identical.

We accomplish this in our Maple code by limiting the recursive construction of circuits. When we are building a circuit with g gates, we only allow the first input to be a subcircuit of between 0 and $\lfloor (g-1)/2 \rfloor$ gates.

Reducing Number of SLP's

We realized one other (admittedly minor) optimization by restricting the inputs to any gate which has zero gates in exactly one of its subcircuits. The input corresponding to the zero-gate subcircuit is not allowed to match either input to the gate producing the other input. If we permitted this situation, which we can think of as having the child of a gate match a grandchild of that gate, we would be able to merge the two gates into one.

A Bridge Too Far

Our initial implementation went too far down this path in one respect. When one or both of a gate's subcircuits had zero gates, we only allowed those inputs to be between 1 and n.

Why is this a problem? This has the effect of restricting the fan-out of gates to 1, meaning that we cannot reuse the output of a gate later. Initially, we did not think this would matter for small circuits, but it does.

A Bridge Too Far

In a 2006 cryptography paper, Markku-Juhani Saarinen included the numbers of 4 variable functions with each circuit complexity. Assuming those numbers to be accurate, some of the functions we indentify as complexity 7 are actually complexity 6.

In theory, fixing this is easy. All we need to do is allow 0-gate subcircuits to be any number lower than the current line number. In practice, the number of circuits increases to an uncomfortable number.

After allowing for gate reuse, the number of 4 variable, 6 gate circuits is now about 42 billion, and the number of 7 gate circuits is about 5.4 trillion.

Circuit Complexity by BEC's

n	0g	1g	2g	3g	4g	5g	6g	7g	Total
1	3								3
2	3	3							6
3	3	3	8	5	3				22
4	3	3	8	34	59	139	130$*$	26$*$	402

The two $*$'ed entries in the table do not agree with Saarinen's enumeration of 4 variable functions. In a future project, we hope to find the additional 6-gate BEC's.

Circuit Complexity by Functions

n	0g	1g	2g	3g	4g	5g	6g	7g	Total
1	4								4
2	6	10							16
3	8	30	114	80	24				256
4	10	60	456	2474	10624	24184	24784$*$	2944$*$	65536

The two $*$'ed entries in the table do not agree with Saarinen's enumeration of 4 variable functions. The first two columns on that line also disagree with Saarinen, but that is merely a difference in definitions. Saarinen defines "NOT x_1" as a gate, and we do not.

Which Functions Are Hardest?

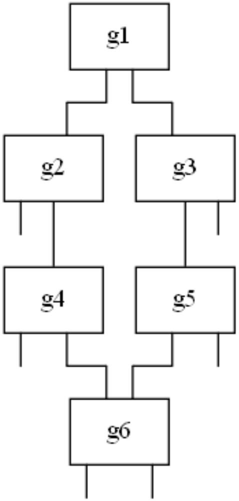
In the 3-variable case, there are 3 BEC's with the maximum complexity of 4 gates. The representative functions of these classes are 22, 23, and 107. Function 22 is true when exactly two of the input variables are true. Function 23 is the majority function (at least two inputs are true). Function 107 is true when exactly one input is true OR its first two inputs are both true.

Taking the majority function as an example, the particular SLP our algorithm found to compute this function is $[[1],[2],[3],[4,1,2],[4,1,3],[1,4,5],[4,1,6]]$. In more traditional notation, this circuit is $x_1 \oplus ((x_1 \oplus x_2) \wedge (x_1 \oplus x_3))$. There are certainly other ways to compute the majority function with 4 gates, but it cannot be computed with 3 or fewer gates.

Future Work

In a future project, we hope to find the discrepancy between our catalog and Saarinen's numbers.

The circuit shown at right represents a particularly challenging case, even with gate reuse available. This circuit effectively has 3 gates on each side.



Monotone Functions

We next turn our attention to monotone functions.

Definition: Let f be a Boolean function on K^n, and let $k \in \{1, 2, \ldots, n\}$. We say that f is <u>positive</u> (respectively, <u>negative</u>) in the variable x_k if $f_{|x_k=-1} \leq f_{|x_k=1}$ (respectively $f_{|x_k=-1} \geq f_{|x_k=1}$). We say that f is <u>monotone</u> in x_k if f is either positive or negative in x_k.

Definition: A Boolean function is <u>positive</u> (respectively, <u>negative</u>) if it is positive (respectively, negative) in each of its variables. The function is <u>monotone</u> if it is monotone in each of its variables.

My first OEIS Sequence!

During this project, I was able to submit my first sequence (A349743) to the OEIS. It is the subsequence of A227723 containing representatives of the big equivalence classes representing functions which are monotone in each of their variables.

Because of the ordering of functions in the original sequence, the BEC representatives are actually positive functions (i.e., positive in each of their variables). One enormous advantage of these functions all being positive is that we only need AND and OR gates (types 1 and 5) to represent them, so the exponential growth of the number of circuits has a factor of 2^g instead of 10^g from choice of gate types.

Advantages of Monotone Functions

In addition to the reduction in gate types required, there are also many fewer BEC's of monotone functions. Specifically for $n = 5$, there are $1,228,158$ BEC's of which only 210 represent monotone functions.

The number of monotone BEC's for n variables is described by the Dedekind numbers (OEIS sequence A003182). Why the Dedekind numbers? Because there is a bijection between antichains and positive Boolean functions. One could specify a positive function by choosing any antichain in the Hamming cube and letting the function's true points be the locus of points in K^n which are members of the antichain with a (possibly empty) subset of the -1's changed to 1's.

Circuit Complexity by BEC's

n	0g	1g	2g	3g	4g	5g	6g	7g	Total
1	3								3
2	3	2							5
3	3	2	4		1				10
4	3	2	4	10	2	6	1	2	30
5	3	2	4	10	26	16	42	35	...
n	8g	9g	10g	11g	12g	13g			Total
5	44	18	3	6		1			210

Again, we would like to confirm these numbers with gate reuse available as a future project.

Introduction

When analyzing the distribution of a combinatorial quantity or random variable X, the statistical moments of X provide information about the shape of the distribution.

In this project, we consider the space of all Boolean functions on n variables. We are particularly interested in the number and sizes of implicants contained within each Boolean function. In the traditional view of Boolean functions, we might write a function as $f(x_1, x_2) = x_1 \vee x_2$. This function has 3 implicants containing 1 point, namely x_1x_2, $x_1\overline{x_2}$, and $\overline{x_1}x_2$. It has 2 implicants with 2 points, x_1 and x_2.

Introduction

When we consider a Boolean function as the subset of points $(x_1, \ldots, x_n) \in \{-1, 1\}^n$ for which the function evaluates to true, implicants are subcubes of the cube which are also subsets of the function.

In the previous example, f would have 3 points (0-dimensional subcubes) and 2 edges (1-dimensional subcubes) contained in the 2-dimensional cube.

We will primarily use the subset and subcube terminology throughout the remainder of this discussion, setting aside that our interest in these objects originally stemmed from the Boolean function interpretation.

Prior Work

Thanatipanonda (2020) uses linearity of expectation and a variation of inclusion-exclusion. He first enumerates all of the combinations of r-subcubes of size k, arranging them by how the subcubes overlap. For example, in the case of $k = 2$ and $r = 1$, we are considering pairs of edges. Two edges can be disjoint, overlap at a point, or coincide.

Using inclusion-exclusion to determine the number of each such type permits a calculation of the moment (in this example $\mathbb{E}[X_1^2]$) by linearity of expectation.

Prior Work

Thanatipanonda produces general formulas for the first two moments of r-dimensional subcubes in functions of n variables.

$$\mathbb{E}[X] = \sum_{i=1}^{\binom{n}{r}2^{n-r}} E[X_i] = \binom{n}{r}2^{n-r} \cdot \frac{1}{2^{2^r}}$$

$$\mathbb{E}[X^2] = \sum_{i=0}^{r} \frac{\binom{n}{i,r-i,r-i,n-2r+i}2^{n-i}}{2^{2^{r+1}}} \cdot \left(2^{2^i} - 1\right) + \frac{\left[\binom{n}{r}2^{n-r}\right]^2}{2^{2^{r+1}}}$$

He also provides a calculation of the third moment for edges. The number of ways the subcubes can overlap grows too quickly for this method to remain practical beyond the third moment.

Data Structure

Every r-dimensional subcube of $\{0,1\}^n$ has the form

$$C = \{(x_1, \ldots, x_n) \in \{0,1\}^n \mid x_{i_1} = \alpha_{i_1}, \ldots, x_{i_{n-r}} = \alpha_{i_{n-r}}\},$$

for some $1 \leq i_1 < \cdots < i_{n-r} \leq n$, $(\alpha_{i_1}, \ldots \alpha_{i_{n-r}}) \in \{0,1\}^{n-r}$.

We use row vectors of length n, in the alphabet $\{0,1,*\}$ to represent subcubes, where $*$ is a wild card. For example, if $n = 7$

and $r = 3$, the 3-dimensional cube

$$\{(x_1, \ldots, x_7) \in \{0,1\}^7 \mid x_2 = 1, x_4 = 1, x_5 = 0, x_7 = 1\},$$

is represented by $*1*10*1$.

Data Structure

We are trying to find a weighted count of ordered k-tuples of r-dimensional subcubes. The natural data structure for these is the set of k by n matrices in the alphabet $\{0, 1, *\}$ where every row has exactly r wildcards.

For any specific, numeric n, there are $(2^{n-r}\binom{n}{r})^k$ of these matrices, and for each and every one of them one can find the cardinality of the union of the corresponding subcubes, let's call it v, and add to the running sum $1/2^v$.

By Linearity of Expectation, this sum is the moment $\mathbb{E}[X_r^k]$.

Kernels and Remainders

A key object in our approach is the **kernel**. Given a $k \times n$ matrix M in the alphabet $\{0, 1, *\}$, we call a column **active** if it contains at least one '$*$'. Note that the matrix has exactly $k \cdot r$ '$*$'s, hence the number of **active columns**, say a, is between r and $k \cdot r$.

We will say that a matrix is in **canonical form** if the active columns are occupied by the a leftmost columns (i.e. its kernel is contiguous starting in the first column). Obviously, there are $\binom{n}{a}$ ways to choose which of the n columns are active, therefore we can compute the contribution to the expectation for the set of matrices in canonical form and multiply by $\binom{n}{a}$.

Kernels and Remainders

We then do a *weighted-count*, where every matrix gets ‘credit’ $1/2^v$, where v is the cardinality of the union of the subcubes represented by the k rows, for the set of matrices in canonical form. Note that there are only finitely many choices for the a leftmost columns.

We divide these into *equivalence classes* obtained by permuting rows and columns and transposing 0 and 1 in any given column. For each equivalence classes, we examine a representative and multiply the weight by the cardinality of the class.

Kernels and Remainders

But what about the $n - a$ rightmost columns? We refer to these submatrices as *remainders*. There are $2^{k(n-a)}$ possible submatrices; there are no wildcards in this region, so the alphabet here is $\{0, 1\}$. Almost all of these have distinct rows, more precisely,

$$\binom{2^{n-a}}{k} k!$$

of them, and these will produce the smallest possible weight in conjunction with any kernel.

The other extreme is that all the rows of the remainder are identical, and then there are only 2^{n-a} choices to fill them in.

Kernels and Remainders

Now for each a and for each set-partition,our Maple code generates the finite set of $k \times a$ matrices in the alphabet $\{0, 1, *\}$. Each of the members of the set partition has its own submatrix, and we ask our computer to find the number of vertices in the corresponding union of subcubes corresponding to each member of the examined set partition. Since they are disjoint, we add them up, getting v for that particular pair (matrix, set-partition), giving credit $1/2^v$.

Moments of Numbers of 1-dimensional Subcubes

Fourth (raw) moment for edges:

$$\mathbb{E}[X_1^4] = \frac{1}{4096}(n^4 2^{4n} + 24n^4 2^{3n} + 144n^4 2^{2n} + 160n^4 2^n + 12n^3 2^{3n} + 48n^3 2^{2n} - 192n^3 2^n + 12n^2 2^{2n} + 48n^2 2^n - 64n 2^n)$$

Fourth central moment for edges:

$$\mathbb{E}[(X_1 - \mu_1)^4] = \frac{1}{1024}(40n^4 2^n - 48n^3 2^n + 12n^2 2^n - 16n 2^n + 12n^4 2^{2n} + 12n^3 2^{2n} + 3n^2 2^{2n})$$

Moments of Numbers of 1-dimensional Subcubes

Fifth (raw) moment for edges:

$$\begin{aligned}\mathbb{E}[X_1^5] = \frac{n^2 2^n}{32768}\big(& n^3 2^{4n} + 40n^3 2^{3n} + 480n^3 2^{2n} + 1760n^3 2^n + 640n^3 \\ & + 20n^2 2^{3n} + 240n^2 2^{2n} - 480n^2 2^n - 3840n^2 \\ & + 60n 2^{2n} + 240n 2^n + 1280n - 320 \cdot 2^n\big)\end{aligned}$$

Fifth central moment for edges:

$$\mathbb{E}[(X_1 - \mu_1)^5] = \frac{5n^3 2^n}{1024}\left(6n^2 2^n + 4n^2 + 3n2^n - 24n + 8\right)$$

Moments of Numbers of 1-dimensional Subcubes

Sixth (raw) moment for edges:

$$\mathbb{E}[X_1^6] =$$
$$\frac{n2^n}{262144}(n^5 2^{5n} + 60n^5 2^{4n} + 1200n^5 2^{3n} + 9120n^5 2^{2n} + 19200n^5 2^n - 14336n^5 + 30n^4 2^{4n} + 720n^4 2^{3n} + 1440n^4 2^{2n} - 29760n^4 2^n - 42240n^4 + 180n^3 2^{3n} + 1440n^3 2^{2n} + 4800n^3 2^n + 30720n^3 - 840n^2 2^{2n} - 2400n^2 2^n + 30720n^2 - 1920n2^n - 53760n + 38912)$$

Moments of Numbers of 1-dimensional Subcubes

Sixth central moment for edges:

$$\mathbb{E}[(X_1-\mu_1)^6] =$$
$$\frac{n2^n}{32768}\cdot\big(120n^5 2^{2n} + 1920n^5 2^n - 1792n^5 - 840n^4 2^n + 180n^4 2^{2n} - 5280n^4 + 90n^3 2^{2n} - 360n^3 2^n + 3840n^3 + 15n^2 2^{2n} - 300n^2 2^n + 3840n^2 - 240n2^n - 6720n + 4864\big)$$

Moments of Numbers of 1-dimensional Subcubes

From these moments, it follows that as n approaches infinity, the third through sixth scaled moments about the mean converge to 0, 3, 0, and 15 respectively. This suggests that the random variable X_1 is asymptotically normal.

Urszula Konieczna (1993) proved the asymptotic normality of X_r in the more general case where each subcube appears with probability $p \in (0, 1)$.

Our approach inspied a new proof of this particular case by Svante Janson (2023) in a joint paper with S. and Zeilberger.

Moments of Numbers of 2-dimensional Subcubes

Third (raw) moment for squares:

$$\mathbb{E}[X_2^3] = \frac{n(n-1)2^n}{2097152}(n^4 2^{2n} + 48n^4 2^n + 576n^4 - 2n^3 2^{2n} + 384n^3 + n^2 2^{2n} + 24n^2 2^n + 1344n^2 - 72n2^n - 1024n - 2176)$$

Third central moment for squares:

$$\mathbb{E}[(X_2 - \mu_2)^3] = \frac{n(n-1)2^n}{32768}(9n^4 + 6n^3 + 21n^2 - 16n - 34)$$

Moments of Numbers of 2-dimensional Subcubes

Fourth (raw) moment for squares:

$$
\begin{aligned}
&\mathbb{E}[X_2^4] = \\
&\frac{n(n-1)2^n}{268435456}(n^6 2^{3n} + 96n^6 2^{2n} + 3072n^6 2^n + 33280n^6 - 3n^5 2^{3n} \\
&\quad - 96n^5 2^{2n} - 1536n^5 + 3n^4 2^{3n} + 48n^4 2^{2n} + 5376n^4 2^n \\
&\quad + 81408n^4 - n^3 2^{3n} - 192n^3 2^{2n} - 10240n^3 2^n \\
&\quad - 53760n^3 + 144n^2 2^{2n} - 5184n^2 2^n - 334848n^2 \\
&\quad + 6976n2^n - 177152n + 15360)
\end{aligned}
$$

Moments of Numbers of 2-dimensional Subcubes

Fourth central moment for squares:

$$\mathbb{E}[(X_2 - \mu_2)^4] = \frac{n\,(n-1)\,2^n}{4194304} \cdot \big(12n^6 2^n + 520n^6 + 12n^5 2^n - 24n^5 + 24n^4 2^n + 1272n^4 - 12n^3 2^n - 840n^3 - 9n^2 2^n - 27n2^n - 5232n^2 - 2768n + 240\big)$$

Moments of Numbers of 3-dimensional Subcubes

Third (raw) moment for cubes:

$$\begin{aligned}
&\mathbb{E}[X_3^3] = \\
&\frac{n(n-1)(n-2)2^n}{1855425871872}(n^6 2^{2n} + 192 n^6 2^n + 10752 n^6 - 6 n^5 2^{2n} - 288 n^5 2^n \\
&\quad + 18432 n^5 + 13 n^4 2^{2n} + 3360 n^4 2^n + 367872 n^4 \\
&\quad - 12 n^3 2^{2n} + 6480 n^3 2^n + 1571328 n^3 \\
&\quad + 4 n^2 2^{2n} - 44592 n^2 2^n + 5206272 n^2 \\
&\quad + 34848 n 2^n + 11860992 n - 17750016)
\end{aligned}$$

Moments of Numbers of 3-dimensional Subcubes

Third central moment for cubes:

$$\mathbb{E}[(X_3-\mu_3)^3] = \frac{n(n-1)(n-2)2^n}{2415919104}(14n^6+24n^5+479n^4+2046n^3 + 6779n^2+15444-23112)$$

Mixed Moments

We also compute a number of mixed moments, such as $E[X_1 X_2]$. These moments are of interest because they provide insight into the correlation between the presence of subcubes of different sizes.

The first method for producing these mixed moments is the brute force enumeration of functions. As in previous cases, this is only feasible up to $n = 4$. Using the Maple program we use to generate the moments previously discussed, we can generate mixed moments up to 10-dimensional subcubes.

Mixed Moments

The first six mixed moments are:

$$\mathbb{E}[X_0X_1] = \frac{n2^n}{16}(2^n+2)$$

$$\mathbb{E}[X_0X_2] = \frac{n(n-1)2^n}{256}(2^n+4)$$

$$\mathbb{E}[X_0X_3] = \frac{n(n-1)(n-2)2^n}{24576}(2^n+8)$$

$$\mathbb{E}[X_1X_2] = \frac{n(n-1)2^n}{1024}(n2^n+8n+8)$$

$$\mathbb{E}[X_1X_3] = \frac{n(n-1)(n-2)2^n}{98304}(n2^n+16n+24)$$

$$\mathbb{E}[X_2X_3] = \frac{n(n-1)(n-2)2^n}{1572864}(n^2 2^n - n2^n + 32n^2 + 64n + 240)$$

Mixed Moments

We generalize Thanatipanonda's formula for second moments to the case of the mixed moment $\mathbb{E}[X_r X_s]$ as follows:

$$\mathbb{E}[X_r X_s] =$$

$$\sum_{i=0}^{\min(r,s)} 2^{r+s-2i} \binom{n}{i, r-i, s-i, n-r-s+i} \frac{2^{n-r-s+i}}{2^{2^r+2^s-2^i}} + \frac{Rest}{2^{2^r+2^s}} =$$

$$\sum_{i=0}^{\min(r,s)} \frac{\binom{n}{i,r-i,s-i,n-r-s+i} 2^{n-i}}{2^{2^r+2^s}} \cdot \left(2^{2^i} - 1\right) + \frac{\binom{n}{r} 2^{n-r} \binom{n}{s} 2^{n-s}}{2^{2^r+2^s}}$$

where $Rest =$

$$\binom{n}{r} 2^{n-r} \binom{n}{s} 2^{n-s} - \sum_{i=0}^{\min(r,s)} \binom{n}{i, r-i, s-i, n-r-s+i} 2^{n-i}.$$

Mixed Moments

As suggested in Thanatipanonda's original paper, we can think of the i in the summation as the dimension of the intersection between two subcubes. This intersection can be no larger than $\min(r, s)$. The multinomial coefficient represents the number of ways to select the i columns with wildcards in both rows, $r - i$ and $s - i$ columns with a wildcard in one row but not the other, and $n - r - s + i$ columns with no wildcards.

This generalized formula computes mixed moments more efficiently than generating the matrices. We are now able to compute mixed moments up to $\mathbb{E}[X_{19}X_{20}]$.

Correlations

Recall that the correlation of two random variables X, Y is:

$$\textbf{Cor}(X, Y) = \frac{\mathbb{E}[XY] - \mathbb{E}[X]\mathbb{E}[Y]}{\sqrt{\left(\mathbb{E}[X^2] - \mathbb{E}[X]^2\right)\left(\mathbb{E}[Y^2] - \mathbb{E}[Y]^2\right)}}$$

Now, having the ability to calculate mixed moments lets us find correlations. It certainly seems reasonable that the correlation between any two sizes of subcube should approach 1 as $n \to \infty$, since functions with more edges will also have more squares, etc. It seems natural to wonder how quickly the correlations approach 1.

Correlations

$$\mathbf{Cor}(X_1, X_2) = 1 - \frac{1}{4n} - \frac{37}{32n^2} + \frac{131}{128n^3} + \frac{115}{2048n^4} - \frac{10543}{8192n^5} + O\left(\frac{1}{n^6}\right)$$

$$\mathbf{Cor}(X_1, X_3) = 1 - \frac{1}{n} - \frac{45}{4n^2} - \frac{79}{4n^3} + \frac{7595}{32n^4} - \frac{21735}{32n^5} + O\left(\frac{1}{n^6}\right)$$

$$\mathbf{Cor}(X_1, X_4) = 1 - \frac{9}{4n} - \frac{1485}{32n^2} - \frac{88509}{128n^3} - \frac{78400125}{2048n^4} + \frac{2327192169}{8192n^5} + O\left(\frac{1}{n^6}\right)$$

$$\mathbf{Cor}(X_1, X_5) = 1 - \frac{4}{n} - \frac{131}{n^2} - \frac{5000}{n^3} - \frac{4357195}{4n^4} - \frac{8037710954}{n^5} + O\left(\frac{1}{n^6}\right)$$

Correlations

It is obvious that the coefficient on $\frac{1}{n}$ must be negative, since the correlation can never exceed 1. More specifically, the coefficient of $\frac{1}{n}$ in $\mathbf{Cor}(X_r, X_s)$ is always $\frac{(r-s)^2}{4}$. The coefficients of lower order terms are more complicated, so we find these by computing $\mathbf{Cor}(X_r, X_s)$ for as many particular pairs (r, s) as possible and analyzing the pattern of the coefficients.

For each $\frac{1}{n^i}$, we treat the matrix of coefficients of that term in $\mathbf{Cor}(X_r, X_s)$ as a function of r and s. We then attempt to find a polynomial which describes that function whose degree in r (respectively s) is at least two less than the number of rows (respectively columns) of the coefficient matrix.

Correlations

$$
\begin{aligned}
\frac{1}{n}: &\quad -\frac{(r-s)^2}{4} \\
\frac{1}{n^2}: &\quad -\frac{(r-s)^2}{32}\left(11r^2+26rs+11s^2-28r-28s+14\right) \\
\frac{1}{n^3}: &\quad -\frac{(r-s)^2}{384}\Big(661r^4+1388r^3s+2166r^2s^2+1388rs^3+661s^4 \\
&\qquad -4380r^3-9012r^2s-9012rs^2-4380s^3 \\
&\qquad +10078r^2+17236rs+10078s^2 \\
&\qquad -9600r-9600s+3256\Big)
\end{aligned}
$$

Correlations

We observe that for each $\frac{1}{n^i}$, the degree of the corresponding polynomial for its coefficients is $2i$.

Each polynomial is uniformly 0 when $r = s$, which we expect because $\mathbf{Cor}(X_r, X_r) = 1$ by definition.

We also note that these polynomials are symmetric in r, s, which must be true because $\mathbf{Cor}(X_r, X_s) = \mathbf{Cor}(X_s, X_r)$.

Definitions

A *combinatorial game* is an open-information game without randomness (in the form of dice or similar devices) in which any current position of the game leads to a finite number of new positions via a legal move by the player whose turn it is. Each player has “perfect information” in the sense that they are aware of the current state of the game and know what moves are available to every player. We will generally refer to the current state of the game as the *position* of the game. Frequently, the goal of the game is to be the last player to make a legal move.

Definitions

A combinatorial game is *finite* if it is guaranteed to produce a winner after some finite sequence of moves. In particular, this means that there can not be any cyclical sequence of moves (i.e. a situation in which the position of the game is identical before and after a non-zero number of moves). In some sense, this is equivalent to saying that the space of eventually reachable positions of the game is strictly decreasing.

Tic-tac-toe is a finite combinatorial game because each move permanently uses one square on the board, so the total number of moves to reach a conclusion cannot exceed the number of spaces. Checkers is not a finite game; if each player has promoted at least one checker to a king, those kings can move between spaces in a manner which repeats a previous position.

Definitions

A combinatorial game is called *impartial* if any move that is legal for one player is also legal for the other.

Chess is a combinatorial game by these definitions, but it is not impartial. All of the pieces are visible, and the possible moves for each piece depend only on the current position, so the perfect information requirement is met. The first player can only move the white pieces, and the second the black, so the same moves are not legal for both of them.

Position Graphs

We can model a finite combinatorial game as a directed graph. The vertices represent positions, and there is an edge from position A to position B if and only if there is a legal move from position A which results in position B. There is one distinguished vertex representing the initial position of the game (e.g. the empty tic-tac-toe board). There can be no cycles if the game is finite, but the underlying undirected graph need not be a tree. It may be possible for multiple sequences of moves to arrive at identical positions.

We will use the notation $\mathcal{N}(A) = \{B_1, B_2, \ldots, B_k\}$ to indicate that from position A, there are legal moves resulting in positions $B_1 \ldots B_k$. If $\mathcal{N}(A) = \emptyset$, there are no legal moves from that position - the current player has lost the game.

Position Graphs

We can classify every position as either a P-position (the player who made the previous move achieving this position will win the game) or an N-position (the player about to make the next move from this position will win the game). These statements about who will win do rely on both players choosing rationally among their available moves. The algorithm for classifying all positions as N or P from the directed graph is as follows:

Position Graphs

1. Label every position V such that $\mathcal{N}(V) = \emptyset$ as a P-position because the next player has no legal move and the previous player has won.
2. For every unlabeled position W, check whether any position in $\mathcal{N}(W)$ is already labeled as a P-position. If so, label W as an N-position because the player about to move can choose to go to a P-position and win.
3. For every unlabeled position X, check whether every position in $\mathcal{N}(X)$ is already labeled as an N-position. If so, label X as a P-position because every possible move for the current player gives the opponent a winning strategy.
4. If any position remains unlabeled, repeat steps 2 and 3.

Sprague-Grundy Values

We define the *minimum excluded number* of a set A, denoted $\textbf{mex}(A)$, as the smallest non-negative integer which is not an element of A. Some examples:

$$\begin{aligned}\textbf{mex}(\{0,1,2,3,5,6,7\}) &= 4\\ \textbf{mex}(\{1,2,3\}) &= 0\\ \textbf{mex}(\emptyset) &= 0\end{aligned}$$

Sprague-Grundy Values

The Sprague-Grundy value of a position V (often called the Grundy value) is defined recursively as follows:

$$\mathcal{G}(V) = \begin{cases} 0 & \text{if } \mathcal{N}(V) = \emptyset \\ \textbf{mex}(\mathcal{N}(V)) & \text{otherwise} \end{cases}$$

The Sprague-Grundy value of a position V is 0 if and only if V is a P-position. The reason for this is that the two ways that $\mathcal{G}(V)$ can be assigned the value 0 are because $\mathcal{N}(V) = \emptyset$ or because $\textbf{mex}(\mathcal{N}(V)) = 0$. In the former case, there is no legal move. In the latter case, every legal move has a nonzero Sprague-Grundy value, so there are no winning options for the current player. Therefore $\mathcal{G}(V)$ includes the information of whether V is a P-postion or an N-position.

Sprague-Grundy Values

Sprague-Grundy values also provide deeper information. One way to see this is in the game Nim. Nim with one nonempty pile is trivial: the current player can remove all of the objects in the pile to win. From the definition above, a position in a single-pile Nim game has a Sprague-Grundy value equal to the number of objects.For single-pile Nim, the Sprague-Grundy value essentially measures how many different mistakes the current player can make.

For Nim with more than one pile, the power of the Sprague-Grundy value becomes apparent. We denote a position in k-pile Nim as $X = (x_1, x_2, \ldots, x_k)$ where x_i is the number of objects remaining in pile i. It is well-known that $\mathcal{G}(X) = \bigoplus_{i=1}^{k} x_i$, i.e. the Sprague-Grundy value of the position is the bitwise XOR of the pile sizes.

Juniper Green

Juniper Green is a two-player impartial combinatorial game invented by teacher Richard Porteous. The basic game uses a board labeled with the integers 1 to n, with a typical value of $n = 100$. The first player chooses any number on the board. The second player then chooses any integer in $[1, n]$ which is either a factor or an integer multiple of the first. The players alternate selecting a factor or multiple of the current number, but they may not choose any number which has been selected previously. Whichever player is left without a legal move loses.

The standard version of the game ("hard" in our Maple package), and the one that is typically played, adds one additional rule: the first player must select an even number.

Direct Approach

The direct method for computing Sprague-Grundy values is with a recursive procedure which

- Computes every available move from the current position P
- Returns 0 if $\mathcal{N}(P) = \emptyset$
- Returns $\mathbf{mex}(\mathcal{N}(P))$ otherwise

This method is very slow for $n > 25$, as the position graph grows exponentially with n.

Winning Initial Positions

For the standard game, Julien Lemoine (2022) has determined whether the first player has a winning strategy for all values of n. He models the game as an undirected graph in which each integer from 1 to n is represented by a vertex, and two vertices are connected if and only if the larger number is a multiple of the smaller number. We will refer to this as an *adjacency graph* to distinguish it from a position graph.

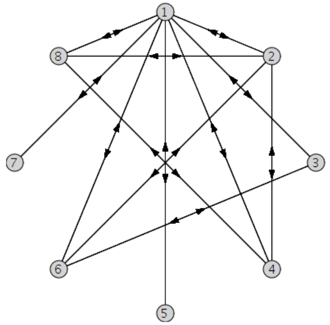
When the next number is selected, the highlighted vertex and its associated edges are deleted from the graph. This method simplifies the analysis because once the graph becomes disconnected, one need only consider the component containing the active vertex.

Pruning the Adjacency Graph

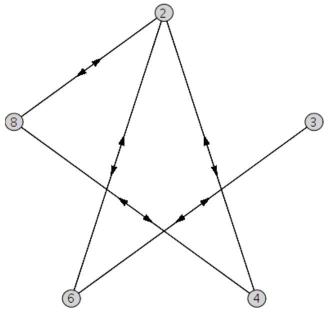
Lemoine also notes that a player selecting 1 loses when the opponent next chooses a prime $p > n/2$. He therefore deletes the vertex 1 and all of the vertices for primes greater than $n/2$ from the adjacency graph, since these are now isolated vertices not available for initial selection.

It seems natural to try this simplification for the computation of Sprague-Grundy values. Unfortunately, pruning the adjacency graph this way does not preserve Sprague-Grundy values, as shown by the following example.

Pruning the Adjacency Graph



$\mathcal{G}(\{2\}) = \mathcal{G}(\{4\}) = \mathcal{G}(\{8\}) = 0$
$\mathcal{G}(\{6\}) = 2$
So $\mathcal{G}(\{\}) = 1$



$\mathcal{G}(\{2\}) = \mathcal{G}(\{4\}) = \mathcal{G}(\{8\}) = 0$
$\mathcal{G}(\{6\}) = 1$
So $\mathcal{G}(\{\}) = 2$

Storing the Adjacency Graph

One improvement which does speed the code somewhat is storing the adjacency table.

Although we still need to iterate through the position graph, we do not need to recompute the legal moves at each step. Instead, we subtract the set of previously selected numbers from the list of factors and multiples of the current number.

Index	Entry
1	$\{2, 3, 4, 5, 6, 7, 8, 9, 10\}$
2	$\{1, 4, 6, 8, 10\}$
3	$\{1, 6, 9\}$
4	$\{1, 2, 8\}$
5	$\{1, 10\}$
6	$\{1, 2, 3\}$
7	$\{1\}$
8	$\{1, 2, 4\}$
9	$\{1, 3\}$
10	$\{1, 2, 5\}$

Results

Using this improvement, we are able to produce the Sprague-Grundy values for $1 \leq n \leq 35$. These values are

$$[0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 2, 2, 2, 0, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 2, 0, 0, 2, 2, 0, 2, 2, 0].$$

The cases above $n = 30$ take a long time to run, even with the improved speed. The exponential growth of the position graph quickly overwhelms any gains. We know from Lemoine's results that the last 0 in the sequence of Sprague-Grundy values appears at $n = 118$, so any attempts to discern periodic behavior would need to extend well beyond that case.

Factor Restriction Variation

At this point, we turn our attention to two new variations of Juniper Green.

In the factor restriction variation, we have two additional game parameters, positive integers a and b. Player 1 starts the game by selecting an even number on the board. Players then alternate selecting unused numbers which are factors or multiples of the current number c, with the additional restriction that the largest permissible divisor is a and the largest permissible multiplier is b. More precisely, the set of legal moves from a postion where c is the current selection is

$$(\{x : x = c/a_1, a_1 \leq a\} \cup \{x : x = c * b_1, b_1 \leq b\}) \setminus \{x : x \text{ has been selected previously}\}$$

Factor Restriction: Symmetric Cases

For the simplest of these cases, $A = B = 2$, we see the following pattern of Sprague-Grundy values:

$$[1, 0, 2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 2,$$
$$2, 2, 2, 2, 2, 3, 3, 3, 3, 3, 3, 3]$$

We further note that the Sprague-Grundy value remains 3 for all $n > 30$, having tested cases up to $n = 600$. This case is trivial in the sense that once player one has made the initial selection, there are only two possible ways the game can proceed: player two can either divide by two or multipy by two. After that point, each player has as most one legal option until one end of the board is reached.

Factor Restriction: Symmetric Cases

For $A = B = 3$, each player is permitted to multiply or divide the current number by 2 or 3. The first 20 Sprague-Grundy values are:

$$[1, 0, 2, 0, 3, 2, 2, 3, 3, 3, 3, 3, 3, 3, 3, 2, 2, 3, 3, 3]$$

Starting at $n = 18$, we see alternating blocks of 3's and 4's. The blocks of 3's are of lengths 9, 22, 71, and 29. The blocks of 4's are of length 5, 10, 25, and 54. We suspect that these blocks will continue to alternate, but we do not have a conjecture for the block lengths.

Factor Restriction: Symmetric Cases

For $A = B = 4$, each player is permitted to multiply or divide the current number by 2, 3, or 4. The first 20 Sprague-Grundy values are:

$$[1, 0, 2, 0, 3, 1, 1, 3, 3, 3, 3, 4, 4, 4, 4, 2, 2, 3, 3, 3]$$

Starting at $n = 18$, we see alternating blocks of 3's and 4's. The blocks of 3's are of lengths 9, 28, and 4. The blocks of 4's are of length 5, 20, and 28. Once again, we suspect that these blocks will continue to alternate, but we do not have a conjecture for the block lengths.

Factor Restriction: Asymmetric Cases

For $A = 2, B = 3$, we see the following pattern of Sprague-Grundy values for $1 \leq n \leq 50$:

$$[1, 0, 2, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3, 3,\\ 3, 3, 3, 3, 3, 3, 4, 4, 4, 4, 4, 3, 3, 3, 3, 3, 3, 3, 3, 3]$$

Here, the blocks of 3's and 4's start at $n = 4$. It appears that the Sprague-Grundy value remains 4 for all $n \geq 162$, which we have confirmed for $162 \leq n \leq 500$.

Factor Restriction: Asymmetric Cases

For $A = 3, B = 2$, we see the following for $1 \leq n \leq 80$:

$$[1,0,2,2,2,2,2,3,3,3,3,3,3,3,3,2,2,2,2,2,\\ 2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,4,4,4,4,4,\\ 4,4,4,4,4,4,4,3,3,3,3,3,3,3,3,3,3,3,3,3,\\ 3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3]$$

The block of 3's at the end of that list runs from $n = 48$ to $n = 127$. It is followed by a block of 4's from $n = 128$ to $n = 335$, after which there is another block of 16 3's, and then a block of 4's which extends at least to $n = 500$.

Additive Variation

In the additive variation, we have two additional game parameters, sets of positive integers A and B. Player 1 starts the game by selecting any number on the board. We do not restrict the initial selection to an even number. Players then alternate selecting previously unused numbers by subtracting an element of A from the current number c or adding an element of B to c. That is, the legal moves from a position where c is currently selected are:

$$(\{x : x = c - a, a \in A\} \cup \{x : x = c + b, b \in B\}) \\ \setminus \{x : x \text{ has been selected previously}\}$$

Additive Variation: Equal Singleton Sets

In the case where A and B each contain the same single integer, say i, the analysis is relatively simple. In all such games, the first player chooses any number c to start the game. The second player then has at most two options: $c - i$ and $c + i$. Neither player has any options after that point and must continue in the same direction. It is clear that the Sprague-Grundy value of the position after the initial selection can be at most 2, since there are only two legal moves from that position. Therefore, we know that the Sprague-Grundy value of the game before the inital selection is at most 3. These cases are simple enough for computation by hand. We will denote the Sprague-Grundy value of the game with n numbers and $A = B = \{m\}$ as $G_m(n)$.

Additive Variation: Equal Singleton Sets

In fact, the sequence of Sprague-Grundy values for $A = B = \{m\}$ is

$$1^m\, 2^{m-1}\, 0\, 2^{m-1}\, \left(2\, 3^{m-1}\, 0\, 3^{m-1}\right)^*$$

where the exponents indicate the number of times a particular Sprague-Grundy value repeats. The $*$ after the final parenthesis indicates that the values inside the parentheses repeat for all subsequent values of n.

Additive Variation: Unequal Singleton Sets

When $A = \{a\}$ and $B = \{b\}$ are distinct singleton sets, the gameplay is somewhat more complicated than the $A = B$ case. One obvious difference is that for any initial selection which is not near 1 or n, player one will have two options at their second turn.

Even so, when there are $\max(a, b)$ consecutive numbers which have already been selected, the remainder of the game will necessarily be played entirely on one side of that group. Effectively, this means that the remaining moves are isomorphic to a game with a smaller value of n where the current selection is near one of the endpoints of the range.

Additive Variation: Unequal Singleton Sets

One other observation is that the sequences of Sprague-Grundy values for $A = \{a\}, B = \{b\}$ and $A = \{b\}, B = \{a\}$ must be identical. This occurs because an initial selection of c by player one in the former case is isomorphic to an initial selection of $n - c + 1$ in the latter case. Therefore, without loss of generality, we will only report results for $A = \{a\}, B = \{b\}$ with $a < b$.

Additive Variation: Unequal Singleton Sets

Sprague-Grundy Values with $A = \{1\}, B = \{b\}, b > 1$:

b	Initial Segment	Cycle
2	$[1, 2, 1, 2, 3, 3, 2, 3, 2]$	$[3]$
3	$[1, 2, 2, 0, 2]$	$[3, 3, 0, 3]$
4	$[1, 2^3, 1, 2, 3^4, 2, 3^3, 2]$	$[3]$
5	$[1, 2, 2, 2, 2, 0, 2]$	$[3, 3, 3, 3, 0, 3]$
6	$[1, 2^5, 1, 2, 3^6, 2, 3^5, 2]$	$[3]$
7	$[1, 2, 2, 2, 2, 2, 2, 0, 2]$	$[3, 3, 3, 3, 3, 3, 0, 3]$

These results suggest two conjectures. For k even, the sequence of Sprague-Grundy values is eventually constant with value 3. For k odd, the sequence of Sprague-Grundy values is periodic, with the cycle consisting of k 3's and a 0.

Additive Variation: Unequal Singleton Sets

Sprague-Grundy Values with $A = \{2\}, B = \{b\}, b > 2$:

b	Initial Segment	Cycle
3	$[1, 1, 2^2, 1, 2^2, 3^4, 2, 3^2, 2]$	$[3]$
4	$[1, 1, 2^3, 1, 2^2, 3^5, 2, 3^3, 2]$	$[3]$
5	$[1, 1, 2^4, 1, 2^2, 3^6, 2, 3^4, 2]$	$[3]$
6	$[1, 1, 2^5, 0, 2^2]$	$[3, 3, 3, 3, 3, 0, 3, 3]$

Additive Variation: Sets of the Form {1,k}

If we set $A = B = \{1, k\}$ for $k > 1$, we see some initial segment of Sprague-Grundy values followed by eventual periodic behavior.

A=B	Initial Segment	Cycle
$\{1,2\}$	$[\,1\,]$	$[0,1]$
$\{1,3\}$	$[\,1\,]$	$[0,2]$
$\{1,4\}$	$[1,0,2]$	$[0,1]$
$\{1,5\}$	$[\,1\,]$	$[0,2]$
$\{1,6\}$	$[1,0,2,0,2]$	$[0,1]$
$\{1,7\}$	$[\,1\,]$	$[0,2]$

Additive Variation: Sets of the Form {2,k}

If we set $A = B = \{2, k\}$ for $k > 2$, we see some initial segment of Sprague-Grundy values followed by eventual periodic behavior as we do in the previous case.

A=B	Initial Segment	Cycle
$\{2, 3\}$	$[1, 1, 2, 0, 1, 0, 3]$	$[0, 1]$
$\{2, 4\}$	$[1, 1, 2, 0, 2]$	$[1, 3, 0, 3]$
$\{2, 5\}$	$[1, 1, 2, 0, 2, 0, 1, 0, 3]$	$[0, 1]$
$\{2, 6\}$	$[1, 1]$	$[2, 0, 2, 2]$
$\{2, 7\}$	$[1, 1, 2, 0, 2, 2, 3, 0, 1, 0, 3, 0, 1, 0, 3]$	$[0, 1]$

Future Work

The first project that presents itself for consideration is extending the additive variation. We would like to verify our conjectures for larger values of n and for a wider variety of sets A and B.

It would also be desirable to find a more efficient way to compute Sprague-Grundy values for the original game in which all factors and multiples are legal moves. It is clear that recursive searching of position graphs will not extend even to $n = 118$ where any search for periodic behavior can begin.

For both of these projects, an auxiliary goal is to generate enough terms of sequences to make them viable candidates for submission to the Online Encyclopedia of Integer Sequences.

Thank you!

Thanks to my advisor, Doron Zeilberger, for all of your advice and encouragement over the past several years.

Thanks to my committee members for your service as such and for the other contributions you have made along the way.

Thanks to my family and friends for all of your support.

Thanks to the entire Rutgers Mathematics Department for being a great environment in which to study, teach, and learn.