Does increasing the dimensions of the grid or increasing the number of lights of the Lights Out puzzle generate more solvable board setups?

Erica Lin

January 29, 2017

Abstract

This paper investigates whether increasing the grid size or increasing the number of light settings in the puzzle Lights Out would generate more solvable solutions. In pursuit of the answer to this question, this essay explores how the board conditions (grid size and number of light settings), along with the game-play functions of toggling and switching lights, can be represented in matrix form. Specifically, this paper delves into how linear algebra techniques—involving the use of Gaussian operations to put matrices into Reduced Row Echelon Form (RREF)—can reveal whether a unique solution exists for a given board setup. Furthermore, this paper examines how the unique solution to any board setup can be found, given that the solution exists. This essay also discusses how the RREF matrix representation of board conditions reveals the proportion of solvable board setups, and how—after finding the total number of board setups—the number of solvable board setups can be found.

This paper investigates Lights Out boards with grid sizes ranging from 2x2 to 9x9 and numbers of light settings ranging from 2 to 5. Ultimately, the findings of this paper are as follows: For the board conditions with fewer light settings (2 or 3), there are a couple of cases where increasing the number of light settings is more effective than increasing the grid size in generating more solvable board setups. However, for the majority of board conditions, increasing the grid size is more effective than increasing the number of light settings in generating more solvable board setups. This outpacing of grid dimensions over number of light settings becomes more drastic as the grid dimensions, as well as the number of light settings, become larger.

Table of Contents

Introduction

Solving puzzles has endured as a beloved pastime for many people. I myself grew up with a deep love for puzzles of all sorts, from jigsaw puzzles, to mazes, to Sudoku. One of my favorite puzzles was a game called Lights Out.

What is Lights Out?

Lights Out was first released by Tiger Electronics in 1995. The game is traditionally played on a 5 by 5 grid of lights, where each light can be in one of two states—on or off. The game starts off with a certain arrangement of on and off lights. The user's goal is to turn off every single light. However, switching a light on or off is not as simple as one might initially assume; toggling a light will change not only the state of the selected light, but those of adjacent lights as well.

So, to clarify, *toggling* a light is not equivalent to *switching* a light. The light that the player selects is *toggled*, which results in the selected light and the adjacent lights to be *switched* on or off.

To illustrate how light toggling works, say the player toggles the center square (*13*) of the Lights Out grid in Figure 1a (the selected toggle has a bolded border). The center square (*13*), along with the squares directly above (*8*), below (*18*), to the left (*12*), and to the right (*14*) are all switched on, resulting in Figure 1b. Say the player then toggles light *19*. Lights *19*, *20*, and *24* would switch on, while lights *14* and *18*—previously on—would switch off, thus producing the configuration show in Figure 1c.

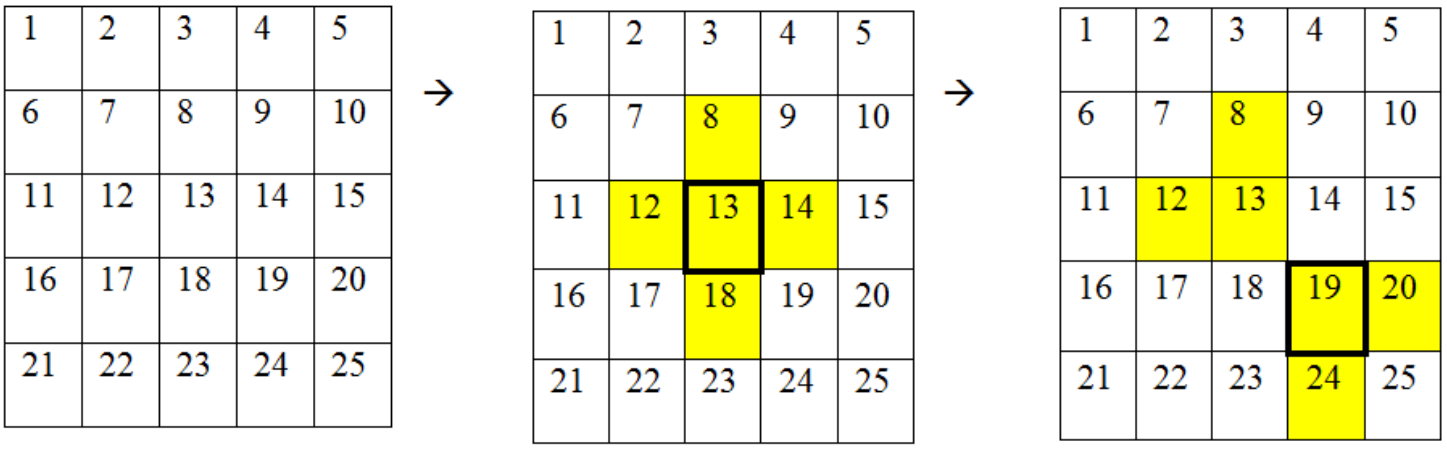


Figure 1 (a) (b) (c)

As a child who delighted in solving puzzles, I also loved to create puzzles. As I played Lights Out, I imagined making the game myself and wondered:

- How do we know whether the setup generated on the Lights Out board is solvable?
- How many solvable puzzles does the traditional Lights Out puzzle with a 5 x 5 grid and 2 light settings give us?
- How much does increasing the dimensions of the board increase the number of solvable setups?
- How much does increasing the number of possible light settings (say, by including multi-colored lights) on the board increase the number of solvable boards?

Ultimately, I wondered, if I were a Lights Out game maker who wanted to release a modified sequel of the game, would increasing the dimensions of the board or increasing the number of light settings create the greater number of puzzles to solve?

Of course, it's not as easy as just calculating the total number of set up combinations given a Lights Out grid with $m \times m$ dimensions and with n number of light settings. The puzzle created must also be solvable.

Despite my avid love for solving puzzles as a young child, I surely did not solve all of the puzzles on my 5 x 5, two light, Lights Out board—I could not even fathom the colossal number of unique puzzles that could be generated by the Lights Out board. However, over the years, I have seen the amazing efficiency and precision with which math can be used to tackle problems, even dealing with unimaginably large quantities. So, at last, I pursue the following question, born from an early curiosity for solving and making puzzles:

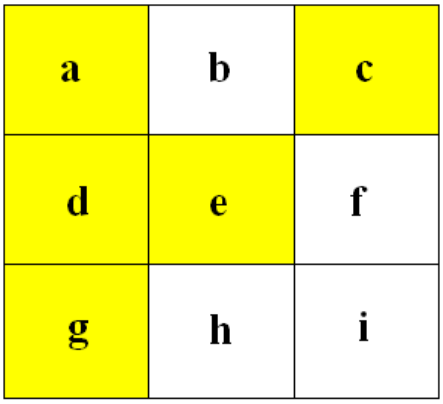
> **Does increasing the dimensions of the grid or increasing the number of lights of the Lights Out Puzzle generate more solvable boards?**

How can we Figure out whether a board set up is solvable?

In order to find the number of solvable boards, we must first find a way to verify that boards have solutions. But is there a single way that would allow us to find the solution to any Lights Out board?

As I researched, I came across a paper describing how linear algebra techniques can be employed to find the most efficient solution to a Lights Out puzzle, by representing the puzzle with matrices [1]. I will explore this concept using an example 3 x 3 board, with two light settings: on (yellow) and off (white) (Figure 2a).

Figure 2a



Before we delve into the matrices, it is important to first note the following: Since the puzzle we are currently dealing with has just two light settings, there is inherent redundancy in the game: switching a light an odd number of times will have the same effect of switching it once, just as switching a light an even number of times will have the same effect of not having switched at all. So, we will be working in modulus 2 with this 2-light-settings board. We would be working in modulus 3 for three settings, modulus 4 for four, modulus 5 for five, and so on.

The Action Matrix

We can use a matrix to represent the action of toggling each light and the effect of each toggle (which lights are affected by a certain toggle)—let us call this matrix the Action Matrix. The Action Matrix for the 3 x 3 Lights Out board is represented below, in Figure 2b:

		Toggle this…								
		a	*b*	*c*	*d*	*e*	*f*	*g*	*h*	*i*
…to switch	*a*	1	1	0	1	0	0	0	0	0
	b	1	1	1	0	1	0	0	0	0
	c	0	1	1	0	0	1	0	0	0
	d	1	0	0	1	1	0	1	0	0
	e	0	1	0	1	1	1	0	1	0
	f	0	0	1	0	1	1	0	0	1
	g	0	0	0	1	0	0	1	1	0
	h	0	0	0	0	1	0	1	1	0
	i	0	0	0	0	0	1	0	0	1

Figure 2b

Each *column* of the 9x9 Action Matrix represents a light that can be toggled. Each *row* represents a light that will or will not be switched. So, if a certain light is toggled, under the respective column head, we have the affected—switched—lights noted with 1's, and the non-affected lights noted with 0's. For example, we can see in Figure 2b that if switch *a* is toggled, only lights *a*, *b*, and *d* will be switched.

The Inverted State Matrix

Figure 2c

$$\begin{bmatrix}0\\0\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} - \begin{bmatrix}1\\0\\1\\1\\1\\0\\1\\0\\0\end{bmatrix} = \begin{bmatrix}1\\0\\1\\1\\1\\0\\1\\0\\0\end{bmatrix}$$

Cleared Board Matrix | *State Matrix* | ***Inverted State Matrix***

The Cleared Board Matrix in Figure 2c represents the states of the lights on a cleared board (all lights are off). The State Matrix represents the initial setup of the lights—in figure 2c, we see the State Matrix for the initial setup in Figure 2a (on lights are represented with 1's and off lights are represented with 0's). Why do we need an Inverted State Matrix? Well, rearranging the equation, we get:

State Matrix + Inverted Matrix = Cleared Board Matrix

From the above equation, we can see that the Inverted State Matrix contains the number of times each light must be switched to change the initial board setup (State Matrix) into the cleared board (Cleared Board Matrix). Thus, the Inverted State Matrix is crucial to finding the solution for a Lights Out puzzle. Please note, since the board in Figure 2a is a Lights Out game with two light settings, we are working in modulus 2, where the only possible values in the State Matrix are 0s and 1s. This brings to attention a fascinating quality unique to the Lights Out game with two light settings. As we see in the first two rows of entries in Figure 2c, working in modulus 2 means that 0 - 1 = 1 (mod 2) and 0 – 0 = 0 (mod 2). Thus, as we work in modulus 2 for a Lights Out games with two light settings, the Inverted Matrix will always be the same as the State Matrix.

Finding the Solution Matrix Using Gaussian Elimination

How do we know which lights to toggle in order to switch each light the appropriate number of times, as shown in the Inverted State Matrix?

Taking the board in Figure 2a as an example, we would need to solve the following equation (Figure 2d):

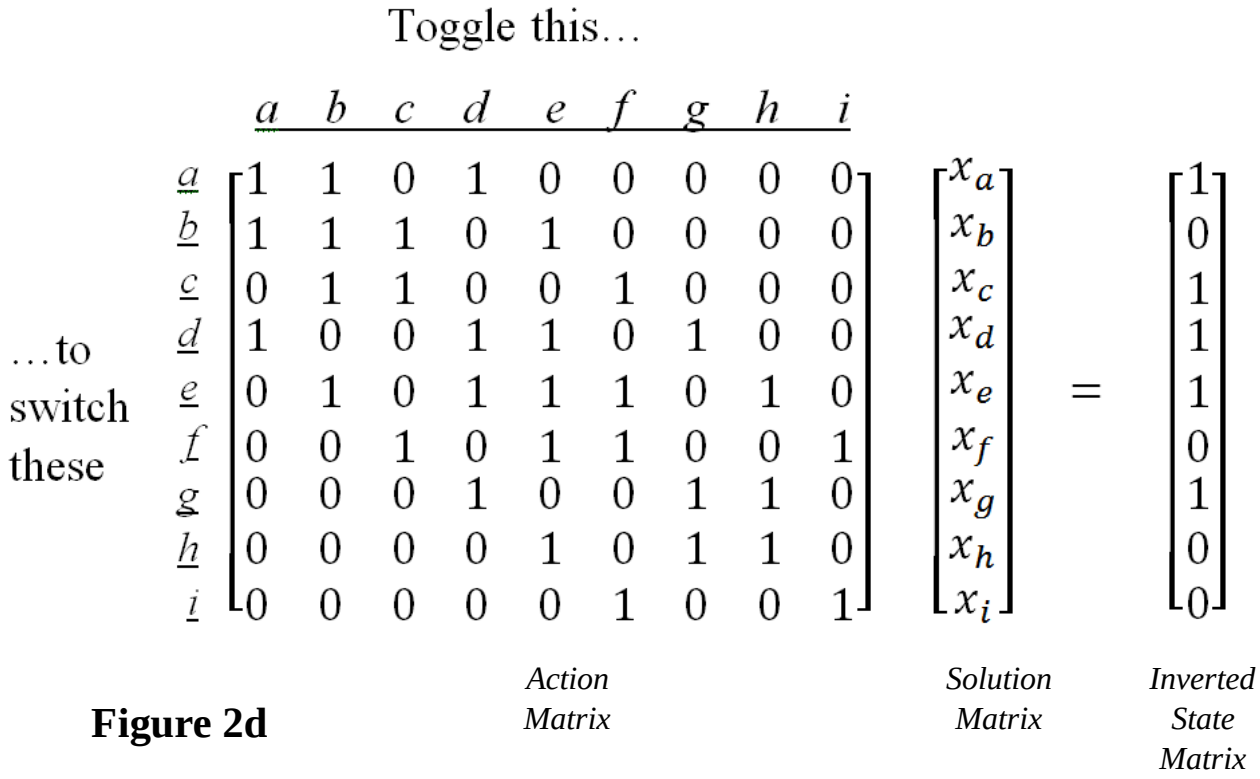


Figure 2d

But why would solving this equation give us the lights that we should toggle? Well, on the left side of the equation, by multiplying the Action Matrix by the Solution Matrix, we are essentially toggling the Solution Matrix lights, while the Action Matrix factors in the number of times each light is switched; this all results to the Inverted State Matrix that we want to achieve. Futhermore, for each initial board setup—there is only one Inverted State Matrix, one cleared board (by definition), and one most efficient solution; thus, all of these can be treated as constants. Taken together, we know that the Solution Matrix is a single, unique matrix that we can solve for in the equation shown in Figure 2d.

To solve for the Solution Matrix, we will need to put the Action Matrix and the Inverted State Matrix into an augmented matrix (Figure 2e).

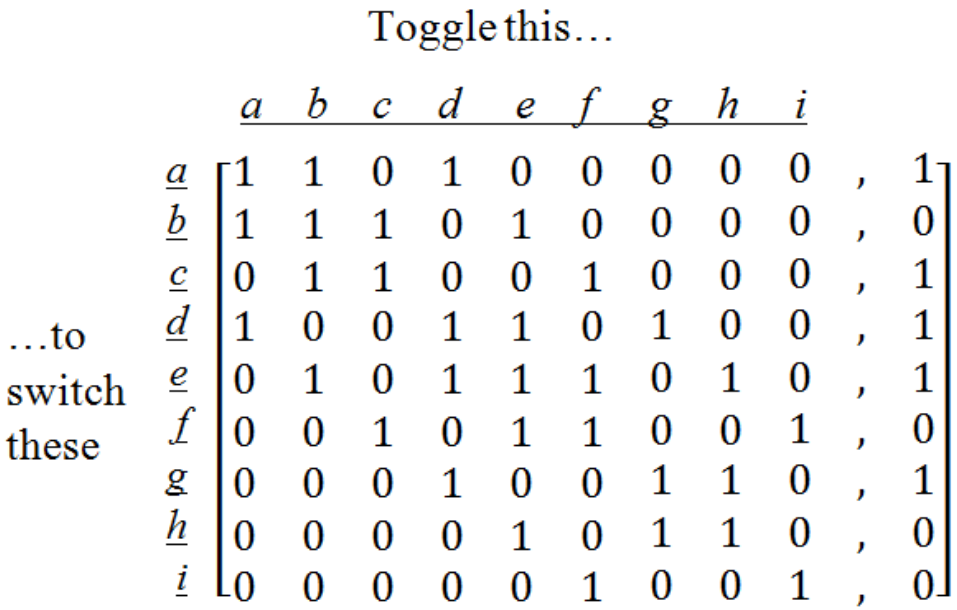


Figure 2e

We then put the matrix into reduced row echelon form (RREF), wherein pivot points of 1's lie on the diagonal extending from the top left corner of the matrix, and the remaining values in the pivot points' columns are all 0's (the RREF of the above matrix can be seen in Figure 2g).

In order to achieve the reduced row echelon form, we can perform any of the following three Gaussian operations:

1) Swap rows: Rj ←→ Ri

2) Replace a row with a multiple of the row: aRi → Ri

3) Replace a row by adding a multiple of another row: Ri + aRj →Ri

In using these Gaussian operations, recall that we are working in modulus 2 for a 2-light-setting board.

Also, I will not use fractions in the Gaussian operations, since we cannot end up with a fraction of a light toggle.

The first operation I perform to put the matrix in Figure 2e into row echelon form is R1 + R2 → R2, transforming Figure 2e to Figure 2f:

Toggle this…

…to switch these

	a	b	c	d	e	f	g	h	i		
a	1	1	0	1	0	0	0	0	0	,	1
b	0	0	1	1	1	0	0	0	0	,	0
c	0	1	1	0	0	1	0	0	0	,	1
d	1	0	0	1	1	0	1	0	0	,	1
e	0	1	0	1	1	1	0	1	0	,	1
f	0	0	1	0	1	1	0	0	1	,	0
g	0	0	0	1	0	0	1	1	0	,	1
h	0	0	0	0	1	0	1	1	0	,	0
i	0	0	0	0	0	1	0	0	1	,	0

Figure 2f

Continuing the process of putting the matrix in Figure 2e into RREF, I use mainly the third Gaussian operation (Ri + aRj →Ri) as I work my way down the augmented matrix, subtracting multiples of the top rows from each of the rows below it so that the values before the pivot point in each row are all 0's. Similarly, I then work my way up, subtracting multiples of the bottom rows from each of the rows above it so that the values after the pivot point in each row are all 0's (See the full list and order of Gaussian operations I use in Appendix 1):

Toggle this…

…to switch these

	a	b	c	d	e	f	g	h	i		
a	1	0	0	0	0	0	0	0	0	,	1
b	0	1	0	0	0	0	0	0	0	,	1
c	0	0	1	0	0	0	0	0	0	,	0
d	0	0	0	1	0	0	0	0	0	,	1
e	0	0	0	0	1	0	0	0	0	,	0
f	0	0	0	0	0	1	0	0	0	,	0
g	0	0	0	0	0	0	1	0	0	,	1
h	0	0	0	0	0	0	0	1	0	,	1
i	0	0	0	0	0	0	0	0	1	,	0

Figure 2g

The right-most column of the rref matrix (Figure 2g) shows us the values of the solution matrix (Figure 2g)—the number of times each light must be toggled as the most efficient solution to the board setup from Figure 2a.

$$\begin{bmatrix} x_a \\ x_b \\ x_c \\ x_d \\ x_e \\ x_f \\ x_g \\ x_h \\ x_i \end{bmatrix} = \left[\begin{matrix} 1 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \end{matrix}\right]$$

Figure 2g

To clear the board, we therefore have to toggle just lights a, b, d, g, and h, each once (and in any order, since only the net number of switches matters in determining the final state of each light).

Are All Setups of a 3 x 3 Dimension, 2 Light Settings Board Solvable?

As I was using Gaussian elimination to find the solution to the board setup in the previous section, I realized that I was only focusing on putting the Action Matrix itself into reduced row echelon form. Since the Action Matrix is universal to all 3 x 3, 2-light-setting Lights Out puzzles, then perhaps I can find a general solution to the 3 x 3, 2-light-setting Lights Out puzzle. If I can indeed find a general solution, that means that *all* board setups in a 3 x 3, 2-light-setting Lights Out puzzle would be solvable.

For the general solution, I will use variables for the Inverted State Matrix in the augmented matrix. Let s_a be the setup state of light a, s_b be the setup state of light b, and so on to light i. (Figure 3a):

Toggle this…

…to switch these

	a	b	c	d	e	f	g	h	i		
a	1	1	0	1	0	0	0	0	0	,	s_a
b	1	1	1	0	1	0	0	0	0	,	s_b
c	0	1	1	0	0	1	0	0	0	,	s_c
d	1	0	0	1	1	0	1	0	0	,	s_d
e	0	1	0	1	1	1	0	1	0	,	s_e
f	0	0	1	0	1	1	0	0	1	,	s_f
g	0	0	0	1	0	0	1	1	0	,	s_g
h	0	0	0	0	1	0	1	1	0	,	s_h
i	0	0	0	0	0	1	0	0	1	,	s_i

Figure 3a

Since the Action Matrix remains constant, I will put the general augmented matrix into reduced row echelon form, using the same steps that I used to put the augmented matrix of the example board (Figure 2a) into reduced row echelon form (Figure 3b):

Toggle this…

…to switch these	a	b	c	d	e	f	g	h	i	
a	1	0	0	0	0	0	0	0	0	$s_a + s_c + s_f + s_g + s_h$
b	0	1	0	0	0	0	0	0	0	$s_e + s_g + s_h + s_i$
c	0	0	1	0	0	0	0	0	0	$s_a + s_c + s_d + s_h + s_i$
d	0	0	0	1	0	0	0	0	0	$s_c + s_e + s_f + s_i$
e	0	0	0	0	1	0	0	0	0	$s_b + s_d + s_e + s_f + s_h$
f	0	0	0	0	0	1	0	0	0	$s_a + s_d + s_e + s_g$
g	0	0	0	0	0	0	1	0	0	$s_a + s_b + s_f + s_g + s_i$
h	0	0	0	0	0	0	0	1	0	$s_a + s_b + s_c + s_e$
i	0	0	0	0	0	0	0	0	1	$s_b + s_c + s_d + s_g + s_i$

Figure 3b

The rightmost column contains the Solution Matrix—the number of times each light should be toggled. To illustrate how the Solution Matrix is found using the expressions shown in Figure 3b, I will once again use the board setup in Figure 2a as an example. The Solution Matrix value for *light a* is found by adding $s_a + s_c + s_f + s_g + s_h$ in modulus 2. Since the setup in Figure 2a shows lights a, c, d, e, g are on, s_a, s_c, and s_h have values of 1 while s_f and s_g (off lights) have values of 0. So, $s_a + s_c + s_f + s_g + s_h$ mod 2 gives us a value of 1 (*light a* must be toggled once)—this corresponds with the value of 1 we had also found for *light a* in Figure 2g.

Most importantly, I want to highlight here that just being able to put the Action Matrix for the 3 x 3, 2 light setting, Lights Out board into reduced row echelon form—and therefore successfully finding a general solution—proves that *every possible setup on a 3 x 3, 2 light setting Lights Out board is solvable*. The most efficient solution for any setup can be found using the expressions found in the right-most column of the reduced matrix in Figure 3b.

Are All Setups of a 4 x 4 Dimension, 2 Light Settings Board Solvable?

Putting the 3 x 3, 2 light setting Lights Out board's augmented matrix into reduced row echelon form by hand was quite time-consuming, so I will use an Excel spreadsheet to manipulate the augmented matrix of the 4 x 4, 2-light setting Lights Out board. My initial augmented matrix looks like this (Figure 4a):

1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	,	s_a
1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	,	s_b
0	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	,	s_c
0	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0	,	s_d
1	0	0	0	1	1	0	0	1	0	0	0	0	0	0	0	,	s_e
0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	0	,	s_f
0	0	1	0	0	1	1	1	0	0	1	0	0	0	0	0	,	s_g
0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	,	s_h
0	0	0	0	1	0	0	0	1	1	0	0	1	0	0	0	,	s_i
0	0	0	0	0	1	0	0	1	1	1	0	0	1	0	0	,	s_j
0	0	0	0	0	0	1	0	0	1	1	1	0	0	1	0	,	s_k
0	0	0	0	0	0	0	1	0	0	1	1	0	0	0	1	,	s_l
0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	,	s_m
0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0	,	s_n
0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	,	s_o
0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	,	s_p

Figure 4a

The furthest that I could manipulate the Action Matrix towards its reduced row echelon form is show below (Figure 4b):

1	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
0	1	0	0	0	0	0	0	0	0	0	0	1	1	0	1
0	0	1	0	0	0	0	0	0	0	0	0	1	0	1	1
0	0	0	1	0	0	0	0	0	0	0	0	1	1	1	0
0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	0
0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	1	0	0	0	0	0	1	0	0	0
0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1
0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0
0	0	0	0	0	0	0	0	0	1	0	0	1	1	1	0
0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	1
0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 4b

We can see that the remaining four columns—which represent the four lights in the bottom row of the 4 x 4 Lights Out grid—could not be put into reduced row echelon form. The rest of the matrix was successfully put into reduced row echelon form. What this all means is that if we were to ignore these four columns (these bottom row lights), there *would* be a general solution for every single board setup. Including these bottom row lights would be problematic in shutting off every single light—unless these bottom row lights were already off in the initial setup. Out of the 16 (2^4) possible initial setups for these four bottom row lights, there is only 1 case where all four bottom row lights are already shut off. *So,* $\frac{1}{16}$ *of all possible board setups are solvable, in the case of the 4 x 4, 2-light-setting Lights Out board.*

Are All Setups of an m x m Dimension, n Light Settings Board Solvable?

Even when using an Excel spread sheet for the Gaussian row operations, it took a long time to put the Action Matrix into row echelon form. Lights Out boards with larger dimensions and more light settings quickly complicate the Action Matrix. For example, just increasing the dimensions of the board from 4 x 4 to 5 x 5 would increase the dimensions of the Action Matrix from 16 x 16—256 elements—to 25 x 25—625 elements. Increasing the number of light settings from 2 to 3 would mean that for each of the 216 elements of the 16 x 16 Action Matrix, I would have to cycle through 3 possible values instead of just 2, as I adjust each element into its row echelon form value.

So, to expedite the process, I used a computer program to put the action matrices of the more complex Lights Out boards into row echelon form. I came across a Javascript program, from Waseda University's professor Kazunori Ueda's Lab, with the code for the Lights Out game and a Check function for the number of "identity patterns" a Lights Out grid has. I downloaded the source code for the program, and I found that the professor's lab defined the number of "identity patterns" as the number of possible combinations of the lights represented by the unsolvable columns, when attempting to put the Action Matrix into reduced row echelon form. The Check function of the code therefore aligns with the method I had used in the previous sections for finding the proportion of solvable board setups; I will use the Check function to find the number of "identity patterns" in the unsolvable columns, and since there is always only one case wherein all lights are already off in the initial setup, the proportion of all possible board setups that are solvable will be $\frac{1}{number\ of\ "identity\ patterns"}$.

I modified the code to also find the identity patterns of Lights Out grids with dimensions 8 x 8 and 9 x 9.

The results are as follows:

Proportion of All Possible Board Setups that are Solvable - THEORETICAL

		Number of Light Settings							
		2		3		4		5	
		Fraction	Percentage	Fraction	Percentage	Fraction	Percentage	Fraction	Percentage
Grid Dimensions	3 x 3	1	100.000%	1	100.000%	1	100.000%	1	100.000%
	4 x 4	$\frac{1}{16}$	6.250%	$\frac{1}{9}$	11.111%	$\frac{1}{256}$	0.391%	$\frac{1}{25}$	4.000%
	5 x 5	$\frac{1}{4}$	25.000%	$\frac{1}{27}$	3.704%	$\frac{1}{16}$	6.250%	$\frac{1}{25}$	4.000%
	6 x 6	1	100.000%	1	100.000%	1	100.000%	1	100.000%
	7 x 7	1	100.000%	1	100.000%	1	100.000%	1	100.000%
	8 x 8	1	100.000%	$\frac{1}{81}$	1.235%	1	100.000%	1	100.000%
	9 x 9	$\frac{1}{256}$	0.391%	$\frac{1}{9}$	11.111%	$\frac{1}{65536}$	0.002%	$\frac{1}{25}$	4.000%

Figure 5a

This method of putting the Action Matrix into reduced row echelon form and then finding the number of combinations of the unsolvable columns, gives me the proportion of board setups that are solvable, at least in theory. However, this theoretical method of finding the proportion of solvable boards is quite lengthy and, I believe, prone to error. Firstly, I am wary that I may have made mistakes in the manual row reduction calculations, especially since they involved so many steps. Also, just a little error in the code that I used for row reduction of the more complex Action Matrices could produce a false proportion.

So, as I tinkered with the Javascript Lights Out game, I also modified the code to add a sampling function, to verify my theoretical proportions. The program can randomly sample 10,000 board setups and tell me how many of the setups produced are solvable. The results are as follows:

Proportion of All Possible Board Setups that are Solvable - SAMPLED - Sample size of 10,000

		Number of Light Settings							
		2		3		4		5	
		Fraction	Percentage	Fraction	Percentage	Fraction	Percentage	Fraction	Percentage
Grid Dimensions	3 x 3	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%
	4 x 4	632 / 10,000	6.320%	1,079 / 10,000	10.790%	Internal Error	Internal Error	387 / 10,000	3.870%
	5 x 5	2,489 / 10,000	24.890%	362 / 10,000	3.620%	607 / 10,000	6.070%	402 / 10,000	6.070%
	6 x 6	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%
	7 x 7	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%
	8 x 8	10,000 / 10,000	100.000%	371 / 10,000	3.710%	10,000 / 10,000	100.000%	10,000 / 10,000	100.000%
	9 x 9	45 / 10,000	0.450%	1,125 / 10,000	11.250%	Internal Error	Internal Error	369 / 10,000	3.690%

Figure 5b

I put the theoretical and experimental results side by side:

Proportion of All Possible Board Setups that are Solvable

		Number of Light Settings							
		2		3		4		5	
		Theoretical	Sample	Theoretical	Sample	Theoretical	Sample	Theoretical	Sample
Grid Dimensions	3 x 3	100.000%	100.000%	100.000%	100.000%	100.000%	100.000%	100.000%	100.000%
	4 x 4	6.250%	6.320%	11.111%	10.790%	0.391%	Internal Error	4.000%	3.870%
	5 x 5	25.000%	24.890%	3.704%	3.620%	6.250%	6.070%	4.000%	6.070%
	6 x 6	100.000%	100.000%	100.000%	100.000%	100.000%	100.000%	100.000%	100.000%
	7 x 7	100.000%	100.000%	100.000%	100.000%	100.000%	100.000%	100.000%	100.000%
	8 x 8	100.000%	100.000%	1.235%	3.710%	100.000%	100.000%	100.000%	100.000%
	9 x 9	0.391%	0.450%	11.111%	11.250%	0.002%	Internal Error	4.000%	3.690%

Figure 5c

The majority of theoretical values correspond with their respective sample values. However, there are three areas of concern, as I have highlighted in the table above:

Firstly, for the 4 x 4 and 9 x 9 grids with 4 light settings, the sampling function had internal errors, so I cannot verify whether my theoretical values were correct, at this current moment. I can go through the program code to find the internal bugs, but for the scope of this investigation though, I can just focus on the data points that I have successfully verified.

Secondly, the 8 x 8 grid with 3 light settings had an experimental value that deviated relatively significantly from the theoretical value. Personally, I believe the sampled value is more reliable, given the sample size of 10,000 and since the other sampled values matched up very closely with the theoretical values. Also, since there were two internal errors, as described above, there likely are bugs with the base program. Just to make sure, [add more as to why the proportion should be 1/27 rather than 1/81]

Finding the Number of Solvable Setups

Since I already have the proportion of possible boards setups that are solvable, my next step would be to find the total number of possible board setups for each Lights Out board. Given that a m x m dimension grid has m^2 number of lights, and each light can have n number of light settings, the total number of possible setups for such a board would be n^(m^2):

Total Number of Possible Board Setups

		Number of Light Settings			
		2	**3**	**4**	**5**
Grid Dimensions	**3 x 3**	2^9 (= $5.120 * 10^2$)	3^9 (~ $1.968 * 10^4$)	4^9 (~ $2.621 * 10^5$)	5^9 (~ $1.953 * 10^6$)
	4 x 4	2^{16} (~ $6.554 * 10^4$)	3^{16} (~ $4.305 * 10^7$)	4^{16} (~ $4.295 * 10^9$)	5^{16} (~ $1.526 * 10^{11}$)
	5 x 5	2^{25} (~ $3.355 * 10^7$)	3^{25} (~ $8.473 * 10^{11}$)	4^{25} (~ $1.126 * 10^{15}$)	5^{25} (~ $2.980 * 10^{17}$)
	6 x 6	2^{36} (~ $6.872 * 10^{10}$)	3^{36} (~ $1.501 * 10^{17}$)	4^{36} (~ $4.722 * 10^{21}$)	5^{36} (~ $1.455 * 10^{25}$)
	7 x 7	2^{49} (~ $5.629 * 10^{14}$)	3^{49} (~ $2.393 * 10^{23}$)	4^{49} (~ $3.169 * 10^{29}$)	5^{49} (~ $1.776 * 10^{34}$)
	8 x 8	2^{64} (~ $1.845 * 10^{19}$)	3^{64} (~ $3.434 * 10^{30}$)	4^{64} (~ $3.403 * 10^{38}$)	5^{64} (~ $5.421 * 10^{44}$)
	9 x 9	2^{81} (~ $2.418 * 10^{24}$)	3^{81} (~ $4.434 * 10^{38}$)	4^{81} (~ $5.846 * 10^{48}$)	5^{81} (~ $4.136 * 10^{56}$)

Figure 6a

The number of solvable board setups is found by multiplying the solvable proportion by the total number of board setups. In my calculations, I will use proportions found theoretically, since these values are likely more accurate:

Number of Solvable Board Setups

		Number of Light Settings			
		2	**3**	**4**	**5**
Grid Dimensions	**3 x 3**	$5.120 * 10^2$	$\sim 1.968 * 10^4$	$\sim 2.621 * 10^5$	$\sim 1.953 * 10^6$
	4 x 4	$4.096 * 10^3$	$\sim 4.783 * 10^6$	--	$\sim 6.104 * 10^9$
	5 x 5	$\sim 8.389 * 10^6$	$\sim 3.138 * 10^{10}$	$\sim 7.037 * 10^{13}$	$\sim 1.192 * 10^{16}$
	6 x 6	$\sim 6.872 * 10^{10}$	$\sim 1.501 * 10^{17}$	$\sim 4.722 * 10^{21}$	$\sim 1.455 * 10^{25}$
	7 x 7	$\sim 5.629 * 10^{14}$	$\sim 2.393 * 10^{23}$	$\sim 3.169 * 10^{29}$	$\sim 1.776 * 10^{34}$
	8 x 8	$\sim 2.418 * 10^{24}$	$\sim 4.239 * 10^{28}$	$\sim 3.403 * 10^{38}$	$\sim 5.421 * 10^{44}$
	9 x 9	$\sim 9.445 * 10^{21}$	$\sim 4.927 * 10^{37}$	--	$\sim 1.654 * 10^{55}$

Figure 6b

Conclusion

So, would it be more efficient to increase the m x m dimensions by an increment of 1 or increase the n number of light settings by an increment of 1, in order achieve a Lights Out board containing the greater number of solvable puzzles? For most cases, especially when using a game with a larger grid and more lights as the reference point, increasing the grid dimensions by 1 increases the number of puzzles at a faster rate than increasing the number of light settings by 1. However, there are a couple of cases, as seen with mostly the boards with lower numbers of light settings (2 and 3 settings), adding to the light settings produce more solvable board setups than adding to the grid dimensions. For example, when expanding from a Lights Out board with a 3x3 grid and two light settings, we see that adding one more light (to three lights) increases the number of solvable board setups from $5.120 * 10^2$ to $1.968 * 10^4$, while increasing the grid dimensions by 1 only increases the number of solvable board setups from $5.120 * 10^2$ to $4.096 * 10^3$. However, in most other cases as shown in the table in Figure 6b, increasing the grid dimensions is more effective than increasing the number of light settings in order to increase the number of solvable board setups.

With regards to the standard Lights Out board released by Tiger Electronics, which has a 5 x 5, two light settings grid, would increasing the grid dimensions or number of light settings generate more solvable boards? Here, we see that is still more beneficial to expand the grid dimensions than to expand the number of light settings. Increasing the grid dimensions by 1, to a 6 x 6 board, produces twice the number of solvable boards ($\sim 6.872 * 10^{10}$ solvable boards) than increasing the number of lights by 1, to a three light settings board ($\sim 3.138 * 10^{10}$ solvable boards).

Also, with larger increments, it becomes clear that it is more efficient to increase the grid dimensions. For example, starting from the 3 x 3 grid with 2 light settings—which has $5.120 * 10^2$ solvable solutions—increasing the number of light settings by 3 units (to 5 light settings) generates just $1.953 * 10^6$ solvable boards while increasing the grid size by 3 units (to a 6 x 6 grid) generates $6.872 * 10^{10}$ solvable boards—that is over 30,000 more solvable board setups generated by increasing grid dimensions rather than the number of light settings. So, overall, increasing the dimensions of the grid seems to produce more solvable boards at a faster rate than increasing the number of light settings.

Some Brief Afterthoughts

Perhaps as a gamemaker, I should not just focus on the *number* of possible solutions, but rather the dimensions/number of light settings that would produce the the highest *proportion* of solvable boards—where there is a higher chance of generating a setup that is solvable. After all, the game console will have to do internal checks to make sure the board setup is solvable before presenting it to the player, and too low of a chance of producing a solvable board setup—such as 0.391% for the 9 x 9, two-light-setting board—may create lag in the game.

References

[1] Barile, Margherita. "Lights Out Puzzle." *Wolfram Mathworld*, mathworld.wolfram.com/LightsOutPuzzle.html. Accessed 5 Sept. 2016.

[2] Feil, Todd, and Marlow Anderson. "Turning Lights Out with Linear Algebra." *Mathematics Magazine*, vol. 71, no. 4, Oct. 1998, pp. 300-03, www.math.ksu.edu/math551/math551a.f06/lights_out.pdf. Accessed 5 Sept. 2016.

[3] Madsen, Matthew A. "Lights Out: Solutions Using Linear Algebra." *Summation*, May 2010, pp. 36-40, cau.ac.kr/~mhhgtx/courses/LinearAlgebra/references/MadsenLightsOut.pdf. Accessed 5 Sept. 2016.

[4] Martín-Sánchez, Óscar, and Cristóbal Pareja-Flores. "Two Reflected Analyses of Lights Out." *Mathematics Magazine*, vol. 74, no. 4, Oct. 2001, pp. 295-304. *JSTOR*, www.jstor.org/stable/2691099. Accessed 5 Sept. 2016.

[5] Missigman, Jennie, and Richard Weida. "An Easy Solution to Mini Lights Out." *Mathematics Magazine*, vol. 74, no. 1, Feb. 2001, pp. 57-59. *JSTOR*, www.jstor.org/stable/2691157. Accessed 5 Sept. 2016.

[6] Torrence, Bruce. "The Easiest Lights Out Games." *The College Mathematics Journal*, vol. 42, no. 5, Nov. 2011, pp. 361-72. *JSTOR*, www.jstor.org/stable/10.4169/college.math.j.42.5.361. Accessed 5 Sept. 2016.

[7] Ueda, Kazunori, and Ueda Lab. "A Lights Out Puzzle with Solver." *A Lights Out Puzzle with Solver*. Waseda University, 2016. Web. 13 Nov. 2016.

Appendices

Appendix 1: Steps to putting the matrix in Figure 2e into row echelon form.

R1 + **R2 →R2**
R1 + **R4 → R4**
R2 ←→ R3
R2 + **R4 → R4**
R2 + **R5 → R5**
R3 + **R4 → R4**
R3 + **R5 → R5**
R3 + **R6 → R6**
R4 + **R6 → R6**
R4 + **R7 → R7**
R5 + **R8 → R8**
R6 ←→ R7
R6 + **R9 → R9**
R7 + **R8 → R8**
R8 + **R9 → R9**
R9 + **R7 → R7**
R8 + **R6 → R6**
R7 + **R4 → R4**
R6 + **R4 → R4**
R6 + **R2 → R2**
R5 + **R3 → R3**
R4 + **R3 → R3**
R4 + **R1 → R1**
R3 + **R2 → R2**
R2 + **R1 → R1**