AM205 Final Project: Modulo-k Lights Out

Shawn Pan and Andrew Ross

December 14, 2016

Introduction

Lights Out is an electronic puzzle game released by Tiger Electronics (the same company that developed the Furby) in 1995. An individual puzzle in Lights Out consists of a configuration of lit and unlit buttons on a 5x5 grid. Pressing any button toggles its state and that of the four adjacent buttons, and the goal is to turn all lights off in as few moves as possible.

The effect of any sequence of presses can be represented as a modulo 2 sum of vectors (where each 2D board coordinate is mapped to a 1D vector position), and because summation is commutative, the sequence of presses will have the same effect regardless of its order. We can represent the constraints of the problem as a system of equations in $\mathbb{Z}_2$ (i.e. modular arithmetic on $\{0, 1\}$), which we can more helpfully write as a matrix equation

$$Ax = b,$$

where $b = \langle g_{11}, g_{12}, \cdots, g_{15}, g_{21}, \cdots, g_{55} \rangle$ is the length-25 vector of grid states g_{ij}, which are 1 if lit and 0 otherwise, x is the length-25 vector of presses, and A is a matrix that encodes the transitions. In the standard Lights Out game, A_{ij} is 1 along the main diagonal and, except at 2D board edges, 1 at positions one and five cells above and below the main diagonal. In the more general case, A can be any matrix encoding transitions, the grid can have any dimensions, and transitions can occur not just in $\mathbb{Z}_2$ (modulo 2) but $\mathbb{Z}_k$.

There are many ways to solve this matrix equation; in the case that A is invertible, we can find its inverse A^{-1} and obtain a unique solution $x = A^{-1}b$. Alternatively, we can use a factorization technique such as the LU decomposition to find x more efficiently. In the case that A isn't invertible, then in general it is possible that there may not be any x such that $Ax = b$, and if there is, then x will no longer be unique (although the shortest x, i.e. the x that minimizes $\sum x_i$, may still be).

[1] provides a great analysis of the core mathematics of the game and several variants. In particular, [1] devotes a significant amount of time to generalizing board dimensions, with some limited discussion of the game in $\mathbb{Z}_3$, which was released as Lights Out 2000. [2] and [3] approach the problem from the perspective of graph theory and graph coloring problems, generalizing it

to arbitrary transition matrices in $\mathbb{Z}_k$. A particular focus of [2] and [3] is determining when the game is always winnable, and on methods of generating families of transition graphs with certain properties.

Our goal is to reproduce and generalize results from [1] and confirm they agree with theorems presented in [2] and [3]. We also want to demonstrate a software package we wrote for solving modular systems of equations and visualizing solutions to Lights Out puzzles.

Additionally, since solutions to light puzzles (arrays of press patterns) can also be interepreted as grids, we consider the recursive problem of solving a grid, then solving its solution, and so on, until we reach a state we have already seen or one that cannot be solved. The simplest version of this (a complete cycle of length 1) corresponds exactly to eigenvectors of the transition matrix with eigenvalue 1. We will explore these cycles of solutions and attempt to relate them to the theoretical work in [1], [2], and [3].

Nullity and Solvability

As we noted above, when A is invertible, then we can solve the puzzle uniquely for any initial board configuration b. When A isn't, we are interested in the dimension of its nullspace. In particular, if the nullspace of A is d-dimensional, then there will be d linearly independent press patterns that have no effect on the board. These are referred to as "quiet patterns" in [1], and a basis for them can be determined by augmenting A with the identity matrix and performing Gaussian elimination to transform A as close to the identity as possible.

The dimension of the nullspace can also be determined by performing modulo-k LU factorization on A, then counting the number of nonzero entries in the upper-triangular matrix U, which is how we calculate it in practice. We do this in Figure 1, which agrees with the tables of results shown in [1] for $k = 2$ and $k = 3$, but generalizes them up to $k = 16$.

We also examined a theoretical result from [3], which states that A must span $\mathbb{Z}_k$ if and only if $\det(A)$ and k are coprime (i.e. their greatest common denominator is 1). Figure 2 shows a plot of those theoretical predictions. Our nullity calculations agree with this theoretical result.

One interesting observation from Figure 2 is that moduli which are powers of two share the same full rank grid sizes, and likewise for powers of three. Furthermore, the set of grid sizes at which A mod 6 is full rank is the intersection of the sets for mod 2 and mod 3, as shown in Figure 3. This result follows directly from the theorem above. An intuitive interpretation is that mod 2 patterns and mod 3 patterns can be simulated in mod 6 by restricting operations to multiples of 3 and 2 presses respectively. Therefore, a non-singular grid size in mod 6 must be non-singular in both mod 2 and mod 3.

Figure 1: Plots of the dimension of the nullspace of the standard Lights Out transition matrix at various grid sizes and moduli

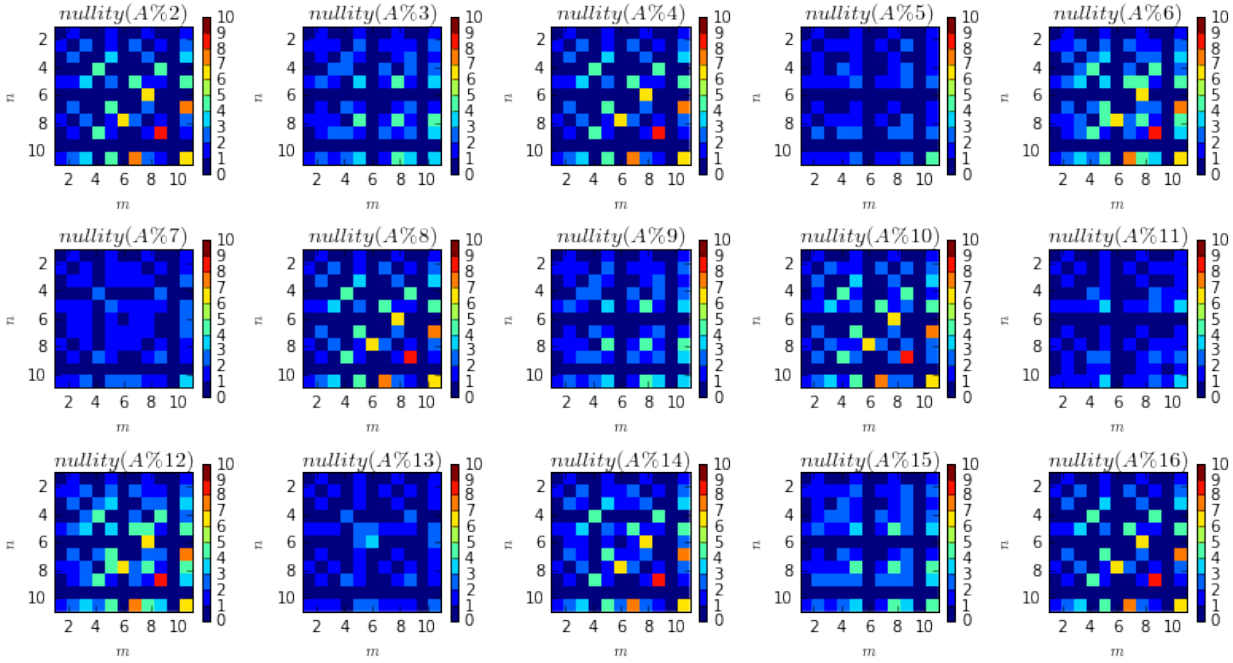


Figure 2: Plots of theoretical predictions for solvability based on [3] The relatively prime entries are in blue.

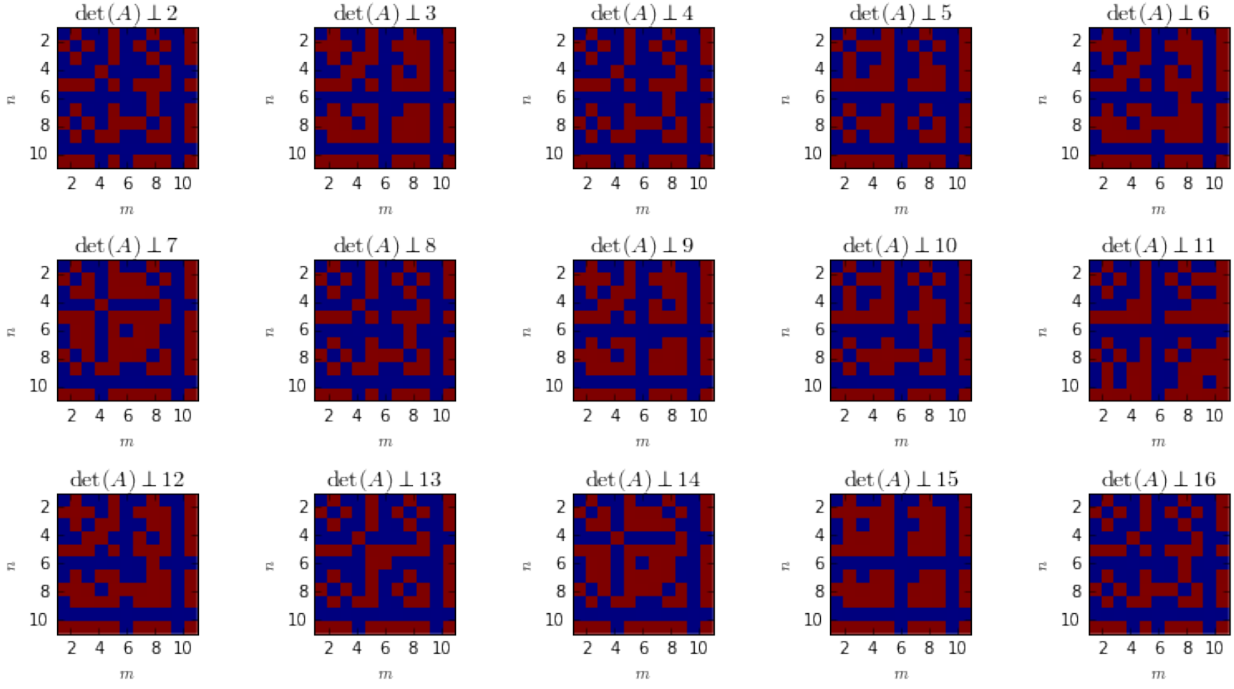
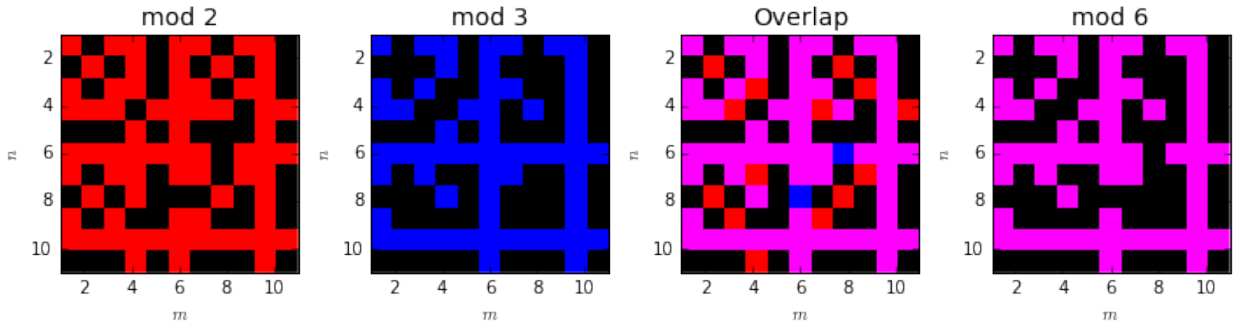


Figure 3: Comparison of full rank grid sizes for the standard transition matrix for mod 2, mod 3, and mod 6



Solving Matrices on a Modular Field

LU Factorization

We have implemented a variant of LU factorization to efficiently solve our transition matrices over a modular field. The standard LU factorization decomposes the matrix into a product of a lower diagonal matrix L and an upper diagonal matrix U. The decomposed form allows problems of the form $Ax = b$ to be solved efficiently chaining two triangular matrix solves. Triangular matrices can be solved with forward and backward substitution.

Since the standard LU algorithm is well known, we will focus our attention on the details for implementing it over a modular field. Addition, subtraction, and multiplication can all be carried on as usual with an additional modulo step at the end to keep the result in the valid range. The major challenge of LU factorization over a modular field that we faced was handling division. Division should be treated as the inverse of multiplication. For example $3/2 = 4$ in mod 5, because $2 \times 4 = 3$. Because we explored relatively small moduli, we precomputed the appropriate division tables by looping over all the pairwise multiplications.

Partial Pivoting

The default LU algorithm quickly runs into issues with zero diagonal entries. Division by zero is undefined during the elimination step. The standard solution to deal with zero entries and to improve numerical accuracy is partial pivoting. Rows are swapped to bring the largest remaining element to the active row, and the final factorization become $PA = LU$ where P is a permutation matrix for the rows. Because all our arithmetic is performed exactly over a finite field, we dont particularly care about getting the largest element for numerical stability. Instead, we just want to get rid of the zero entry on the diagonal.

A naive first attempt is to swap a row with zero diagonal entry with any row containing a non-zero entry in the corresponding spot. This method works perfectly well for a prime modulus, but we run into issues with composite moduli. For example, consider $1/2$ mod 6. There is no solution satisfying $2x = 1$ in mod 6. This undefined division becomes a problem if 2 is a diagonal

entry. Therefore, we require that the pivots values for the diagonal be relatively prime to the modulus.

As a second pass, we use partial pivoting to select a row to make the diagonal entry relatively prime to the modulus. We implemented Euclid's algorithm to find the greatest common factor. It turns out that this solution works when the modulus is prime or a power of a prime. The solution solves many matrices with other composite moduli, but runs into some issues. Consider the following example of an invertible matrix in mod 6:

$$A = \begin{pmatrix} 2 & 3 & 0 \\ 3 & 1 & 1 \\ 3 & 0 & 2 \end{pmatrix} \quad A^{-1} = \begin{pmatrix} 2 & 0 & 3 \\ 3 & 4 & 4 \\ 3 & 3 & 5 \end{pmatrix}$$

1 and 5 are the possible pivot factors which are relatively prime to 6. None of the entries in the first column of A are suitable as pivot factors. Intuitively, the problem is that none of rows are suitable for pivoting, but the matrix is not singular because a linear combination of 2 rows is a valid pivot: $2 + 3 = 5$. There are however several 1s in the middle column, which would be valid pivots if we could access them.

Full Pivoting

We next implemented full pivoting to get more choices for possibly pivot factors. With full pivoting, columns may also be swapped. The factorization then becomes $PAQ = LU$, where P and Q are the row and column permutation matrices respectively. For the matrix in the previous example, full pivoting allows us to move the 1 in the center of the matrix to the active diagonal spot for factorization. Although full pivoting is sufficient for over 90% of the cases we explored, occasionally it still fails to solve an invertible matrix, such as the following matrix in mod 6:

$$A = A^{-1} = \begin{pmatrix} 4 & 3 \\ 3 & 4 \end{pmatrix}$$

Checking for Linear Combinations

To deal with the matrix above, we need to check for linear combinations of rows and columns that may create suitable pivot values. In our example, $4 + 3 = 1 \bmod 6$, so adding the two rows will make the matrix solvable. Systematically checking all the linear combinations is complex and computationally expensive, as the direct approach is exponential with the number of remaining rows. We instead take a stochastic approach and add 100 random rows in an attempt to create a linear combination with a suitable pivot. If no suitable pivot is found, we treat the matrix as singular. In practice this approach solved all the matrices we tried for our experiments. It may be interesting future work to look into efficient deterministic ways to determine the correct linear combination of rows to take.

It is somewhat surprising that full pivoting alone is not sufficient to solve all non-singular matrices. An interesting side effect of taking linear combinations of rows is that the row permutation

matrix P is no longer strictly a permutation matrix. P may include extra values representing the linear combinations taken.

Singular Transition Matrices

Although LU factorization followed by forward and backward substitution works well for solving full rank matrices, the default procedure does not work for singular matrices. When exploring the light out problem, we often run into this issue. For example, the transition matrix for the standard 5x5 lights out game has rank 23 and nullity 2.

Multiple Solutions for Prime Modulus

The rank of the matrix is equal to the number of non-zero entries along the upper diagonal factor U. Therefore, we may proceed as usual with the standard LU algorithm until the backward substitution step. In the backward substitution step, we will run into n zero diagonal entries, where n is the nullity of the original matrix. These zero diagonal entries correspond to equations of the form $0x = 0$ or $0x = 1$. In the $0x = 1$ case, the system is inconsistent and has no solutions. In the $0x = 0$, the system has multiple solutions, because we essentially have an unconstrained free parameter.

Let $z_1, z_2, \ldots, z_n$ be the indices for the singular diagonal entries, i.e. $U_{z_i,z_i} = 0$. We can change the matrix to be no longer singular by setting $U_{z_i,z_i} = 1$, essentially adding an extra equation to the system. We do not have any constraint on x_{z_i} and can set it to any of value from 0 to $m - 1$ for a field with prime modulus m. We consider 3 different ways to picking the free parameter x_{z_i}. Our solve method implementation takes in a flag for which method to use.

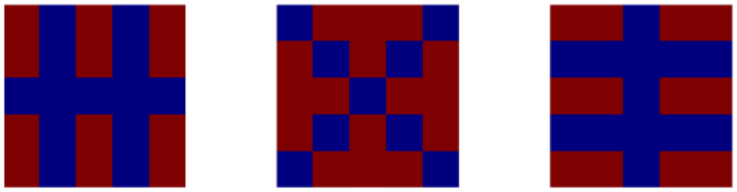
- Any: We pick all x_{z_i} to be 0, which will return an arbitrary solution.
- Basis: For each of the n zero diagonal entries, we perform a reverse solve with a single x_{z_i} set to 1 and all other free parameters x_{z_j} $(j \neq i)$ set to 0. This procedure finds n solutions which form a basis for the nullspace.
- All: For each of the m^n possible free parameter choices for x_{z_i}, we perform a reverse solve. This procedure finds all possible solutions to the equation. The number of solutions grows exponentially with the dimensions of the nullspace.

Applications of Finding Multiple Solutions

The standard 5x5 lights out game has been studied in depth by [1]. The transition matrix for the standard 5x5 lights out game has rank 23 and nullity 2. The interpretation is that not all states in the game are reachable from a blank grid. The game has 2^{23} solvable states, each with 2^2 solutions. Our singular solver can be applied in this situation to find all the solutions for a 5x5 puzzle state.

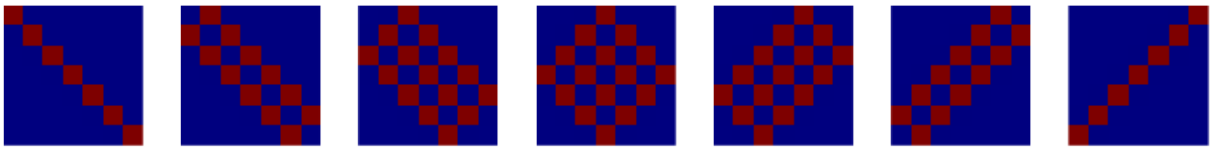
Furthermore, we can extract the quiet states for the 5x5 case by finding all the solutions to $Ax = 0$. Quiet states are members of the null space and represent a series of presses with no effect on the pattern. As shown in Figure 4, our results match previous work.

Figure 4: Plots of all quiet states for the standard 5x5 game, which match the results from [1].



Another application of finding multiple solutions to a singular matrix is to extract eigenvectors. We can find eigenvectors of eigenvalue λ by solving the singular matrix $A - \lambda I$. For the 7x7 mod 2 case, we find 7 linearly independent eigenvectors as shown in Figure 5. After discarding the all zeros case, there are 127 total eigenvector patterns. In mod 2, the eigenvectors have a nice interpretation that the presses required to solve a pattern are exactly the pattern itself.

Figure 5: Plots of the 7 basis eigenvectors of the standard 7x7 game (mod 2).

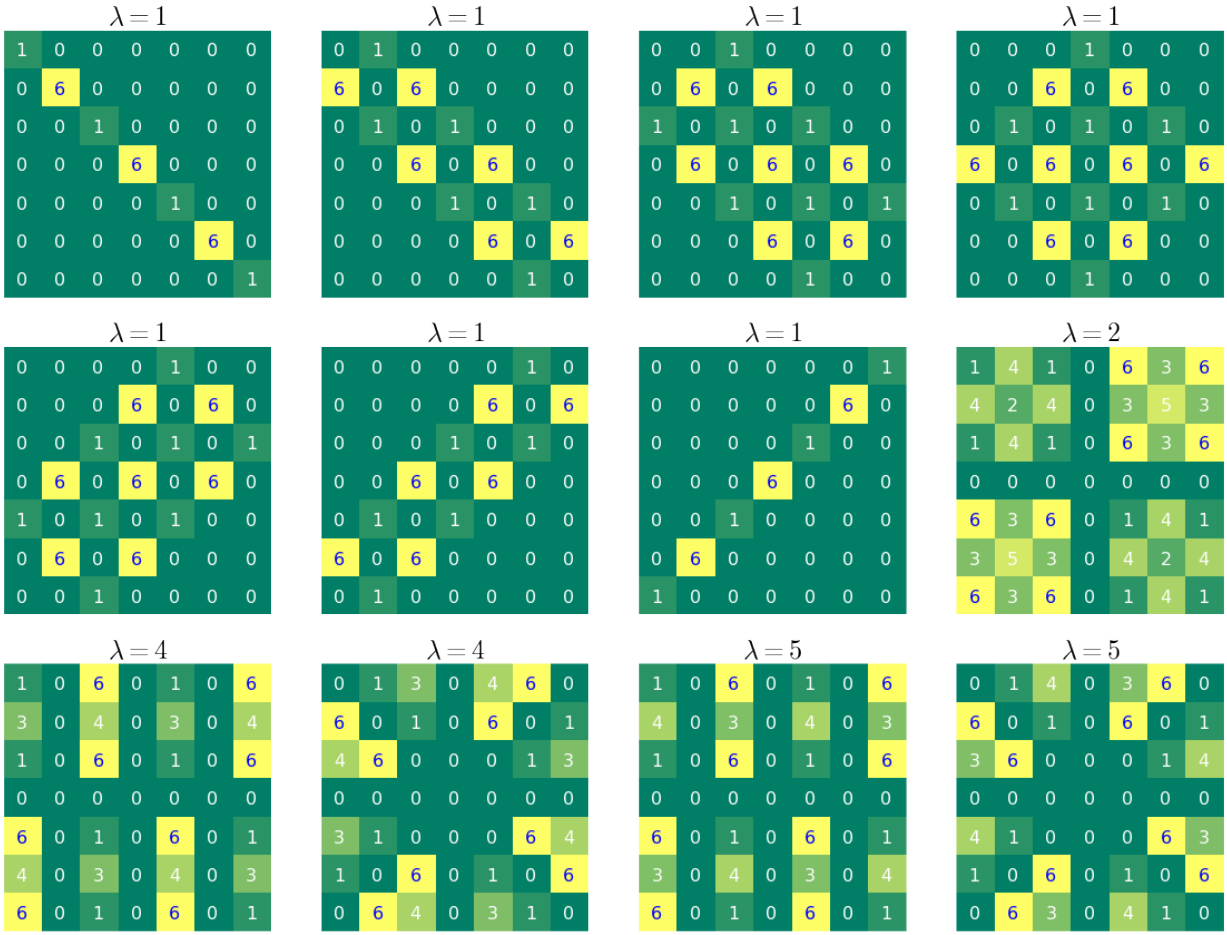


For a higher modulus such as 7, we can get eigenvectors corresponding to different eigenvalues as shown in Figure 6. The presses required to clear the board are $-\lambda^{-1}$ times the eigenvector mod 7. Basically, you would need to press the buttons of the eigenvector multiple times to cycle around to a blank board.

Challenges with Composite Moduli

We briefly explored finding solutions to singular matrices over fields with composite moduli. The rank of a matrix would now be the number of relatively prime diagonal entries in U. A major challenge is that not all singular values are equivalent, and we have a wide variety of cases beyond $0x = 0$ and $0x = 1$. For example, in mod 6 you can have rows representing equations such as $3x = 0$ which has 3 solutions of 0, 2, and 4, or $2x = 1$ which has no solution. Even to correctly count the number of solutions is challenging and depends on the particular non-relatively prime values along the diagonal. Enumerating all the solutions for the composite case may be interesting future work.

Figure 6: Plots of the basis eigenvectors of the 7x7 grid (mod 7).



Eigenvectors and Solution Cycles

Eigenvectors with eigenvalue 1 are characterized by the following equation:

$$Ax = x,$$

where, starting from an unlit board, pressing all of the buttons in x that number of times will produce a board lit in the exact same pattern (or we can solve the board by pressing $-x \bmod k$). We can also consider situations where pressing each button in x, then pressing all of the buttons in the resulting pattern, and so on for p iterations will eventually reproduce x. We'll refer to this case as a cycle of period p. These cycles are characterized by:

$$A^p x = x,$$

or in other words, x is an eigenvector of A^p with eigenvalue 1.

For every non-singular A (at least for standard Lights Out transition rules), we found that $A^{p'}$ actually always equals the identity matrix for some p' (although there is still a spectrum of eigenstates of lower powers of A), so that *every* possible grid state is actually part of a solution cycle for some $p \leq p'$. This result suggests an interesting way of solving a lights out pattern for a non-singular grid size. By repeatedly pressing all the lit up buttons, eventually one clears the board because $A^{p'-1} = A^{-1}$. For the standard 7x7 game, $p' = 8$ and any initial state can be solved by doing 7 passes at pressing all the lit up buttons. We will call p' the identity root of A.

Figure 7 shows counts of the numbers of button states that belong to cycles of different periods p for 2×2 and 3×3 grids at several $\mathbb{Z}_k$. The number of distinct cycles, rather than the number of states, can be obtained by dividing the cell's value by p. Note that generating these quantities requires exhaustively iterating through every possible grid (of which there are k^{mn}), so they can only be computed iteratively when at least one of k, m, and n is relatively small. Some patterns in the data are already immediately evident.

Figure 7: Counts of eigenstates of A^p for 2×2 and 3×3 grids at various $\mathbb{Z}_k$

Number of states in cycles of period p on the 2×2 grid						
k, p	**1**	**2**	**4**	**6**	**8**	**Total**
2	4	12	0	0	0	$16 = 2^4$
3	9	18	0	0	0	36
4	16	112	128	0	0	$256 = 4^4$
5	25	100	500	0	0	$625 = 5^4$
6	36	396	0	0	0	432
7	49	294	0	2058	0	$2401 = 7^4$
8	64	960	1024	0	2048	$4096 = 8^4$
9	81	648	0	0	0	729
10	100	1900	8000	0	0	$10000 = 10^4$

Number of states in cycles of period p on the 3×3 grid							
k, n	**1**	**2**	**4**	**8**	**12**	**24**	**Total**
2	8	24	480	0	0	0	$512 = 2^9$
3	27	0	0	19656	0	0	$19683 = 3^9$
4	64	960	130048	131072	0	0	$262144 = 4^9$
5	125	0	0	0	78000	1875000	$1953125 = 5^9$

We also iterated through and computed the identity root p' (again, the power at which $A^{p'}$ becomes the identity) for many grid sizes and moduli. Some numerical results are shown in Figure 8.

Figure 8: Identity root for $\mathbb{Z}_2$ and $\mathbb{Z}_3$ at various grid sizes

Identity roots in $\mathbb{Z}_2$ by grid size						
$n \times m$	**2**	**3**	**4**	**5**	**6**	**7**
2	2					
3	-	4				
4	6	12	-			
5	-	-	12	-		
6	14	28	126	28	14	
7	-	8	24	-	56	8

Identity roots in $\mathbb{Z}_3$ by grid size						
$n \times m$	**2**	**3**	**4**	**5**	**6**	**7**
2	-					
3	-	8				
4	8	-	-			
5	-	-	24	-		
6	26	728	728	78	26	
7	-	80	-	-	-	80

For singular matrices A, we also found cases where

$$\begin{aligned} A^{p_1+p_2}x &= A^{p_1}x, \\ A^p x &\neq x \text{ for all } p, \end{aligned} \tag{1}$$

that is, where after p_1 iterations of applying A to an initial state x, we enter a repeating cycle with period p_2 that never returns to any members of our initial set of p_1 transient states. In these cases, $A^{p'}$ never equals for any p' the identity, which must be true for x to be transient. Some examples are visible in Figures 9 and 10.

Figure 9: 3-state transient path ending in a 6-state cycle of A for $\mathbb{Z}_6$ on a 5×5 grid.

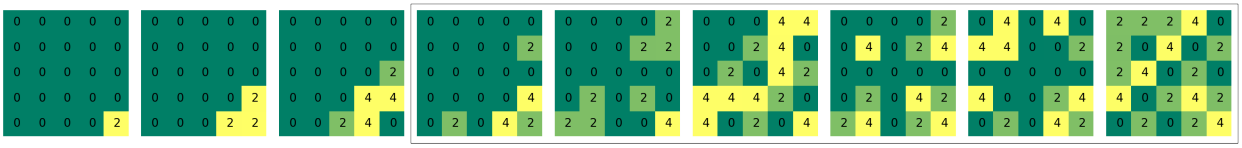
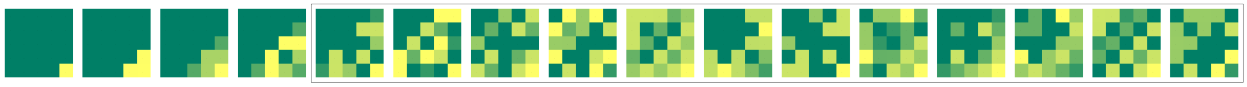


Figure 10: 4-state transient path ending in a 12-state cycle of A for $\mathbb{Z}_6$ on a 5×5 grid.



There is also interesting structure in the numbers of transient states. For example, for A on the 2×2 grid mod 6, for each of the 36 eigenvectors of A, there are two states that transiently lead to it, and similarly for the 396 eigenvectors of A^2. Altogether, that accounts for every $36 + 72 + 396 + 792 = 6^4$ possible grid in 2×2 $\mathbb{Z}_6$. More complex patterns occur at higher grid dimensions and modularities, including cases where states that transiently lead to eigenstates of A^p *themselves* have states which transiently lead to them. For example, on the 2×2 grid mod 9, each of the 81 eigenstates of A has 2 immediate transient parents, which themselves each have 3 parents. The same pattern holds for the 648 eigenstates of A^2, and again we obtain the full state space.

We can summarize and analyze these results by considering how A implicitly organizes the k^{mn} button grid states into a graph (where you transition between states by applying A). In the case when A is invertible, the state space is entirely divided into disconnected rings, with k^n tiny rings of size 1 that consist of (not necessarily independent) eigenstates of A, and more rings that correspond to eigenstates of A^p. When A is singular, there still appear to be disconnected rings of eigenstates of A^p (and there are still k^n eigenstates of A), but they do not constitute the entirety of the state space, and instead the rest of the states form trees branching off of each ring state, seemingly with equal branching factor everywhere (although we only considered a few examples). Future work would ideally analyze this structure theoretically, rather than experimentally.

Generating Solution Videos

We also explored methods of visualizing solutions to these variants of Lights Out, including automatically generating video using sequences of matplotlib figures and ffmpeg to join them together. We also experimented with transforming images into grids for solving. Note that since the order of button presses is arbitrary, each grid can be solved in whatever manner is most visually appealing. Frames from several examples are shown below in Figures 11 and 12.

Figure 11: One of many intermediate frames from a solution video to a 64×64 grid in $\mathbb{Z}_{51}$ with a complicated transition matrix A_c. A full video is available at goo.gl/z2jQuk. Note that since A_c is invertible, the sequence of presses on the right can be transformed back into the original image, which may be of cryptographic interest.

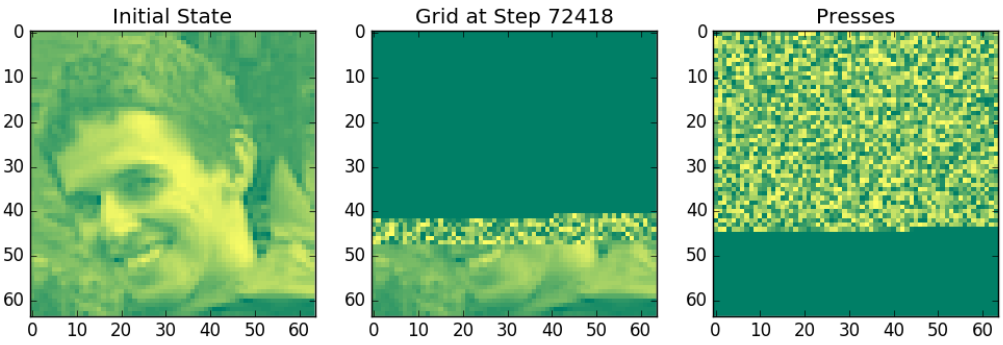
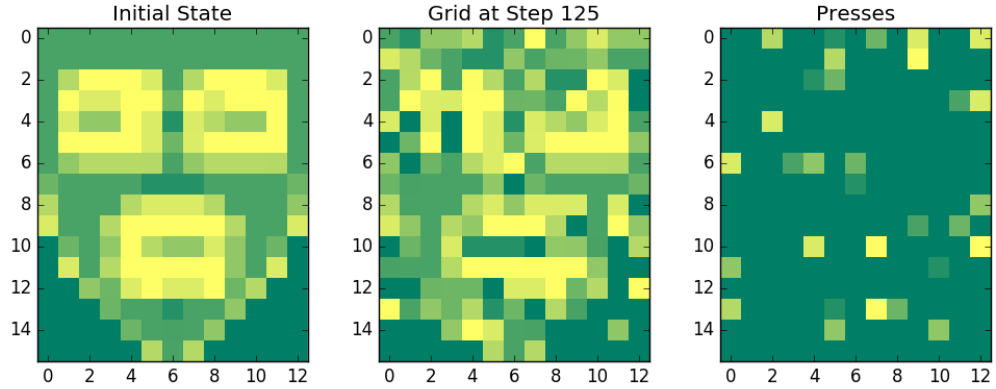


Figure 12: An early frame from a solution video where the presses occur in a random order. Full video available at goo.gl/rT1z23.



Conclusion

We have explored the generalized version of the Lights Out game over higher moduli and various grid sizes. We have implemented numerical algorithms to efficiently solve and analyze matrices over a modular field. We have found methods of extracting multiple solutions out of singular matrices, which have applications in extracting quiet states and eigenvectors. We have characterized the cyclical nature of the grid states. Our techniques work best for prime moduli, but we have made significant progress into dealing with composite cases.

References

[1] J. Scherphuis, *The Mathematics of Lights Out*, http://www.jaapsch.net/puzzles/lomath.htm. Accessed December 2016.

[2] A. Giffen and D. B. Parker, *On Generalizing the "Lights Out" Game and a Generalization of Parity Domination*, preprint, 2009. Available at http://faculty.gvsu.edu/parkerda/profstuff/papers/hyperlogpd.pdf.

[3] S. Edwards, V. Elandt, N. James, K. Johnson, Z. Mitchell, and D. Stephenson, *Lights Out on finite graphs*, Involve 3 (2010), 17-32. Available at http://msp.org/involve/2010/3-1/involve-v3-n1-p03-s.pdf.