AM205 Final Project: Modulo-n Lights Out

Shawn Pan and Andrew Ross

December 14, 2016

Introduction

Lights Out is an electronic puzzle game released by Tiger Electronics (who notably also 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

$$Ax = b, \tag{1}$$

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 $k = 13$.

What's interesting, though, is that our plots in Figure 1 actually disagree in certain cases with a theoretical result from [3], which states that A must span $\mathbb{Z}_k$ if $det(A)$ and k are coprime (i.e. their greatest common denominator is 1). Figure 2 shows plot of those theoretical predictions, which you can see agree with the results in Figure 1 but only when k is prime. Coincidentally, [3] only proves that result for prime values of k, but it is unclear whether their result actually doesn't hold for non-prime values of k (or if there is an error in our code).

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

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

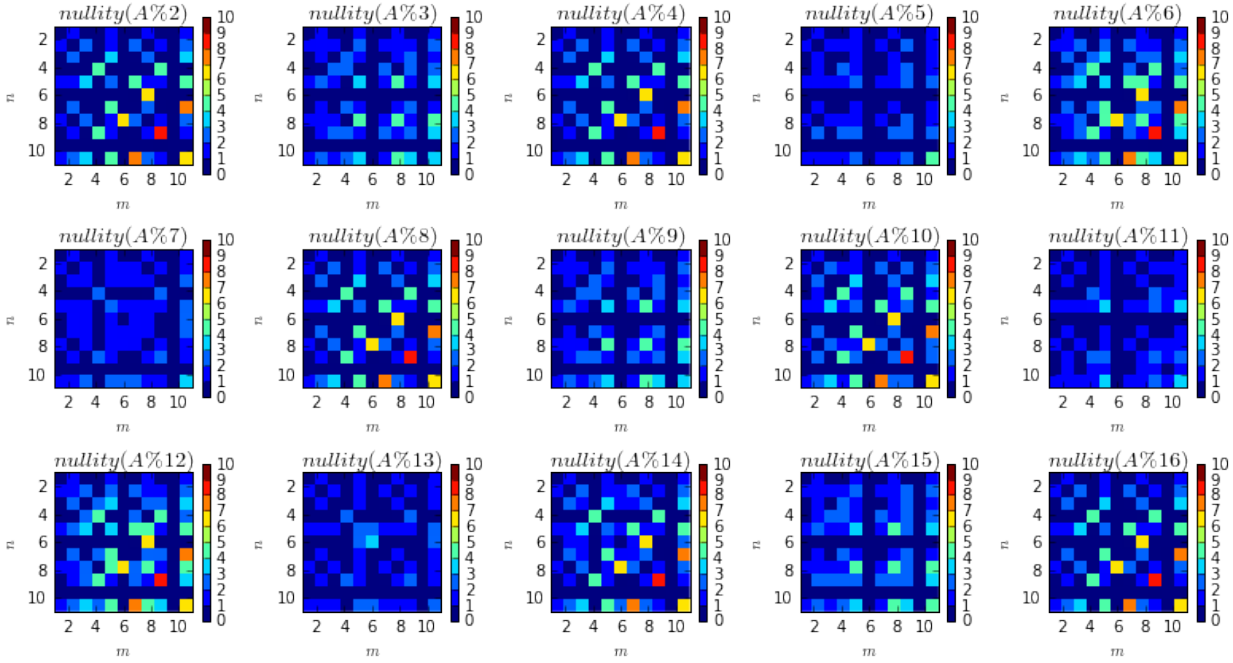


Figure 2: Plots of theoretical predictions for solvability based on [3] The relatively prime entries are in blue.

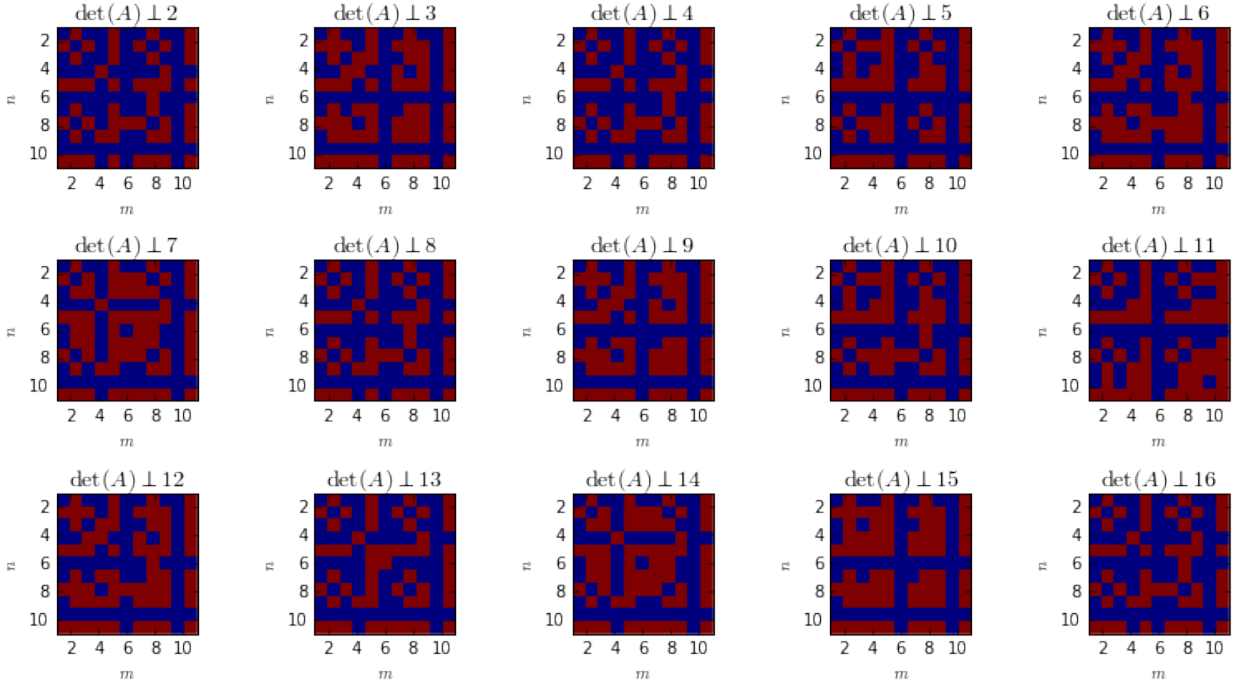
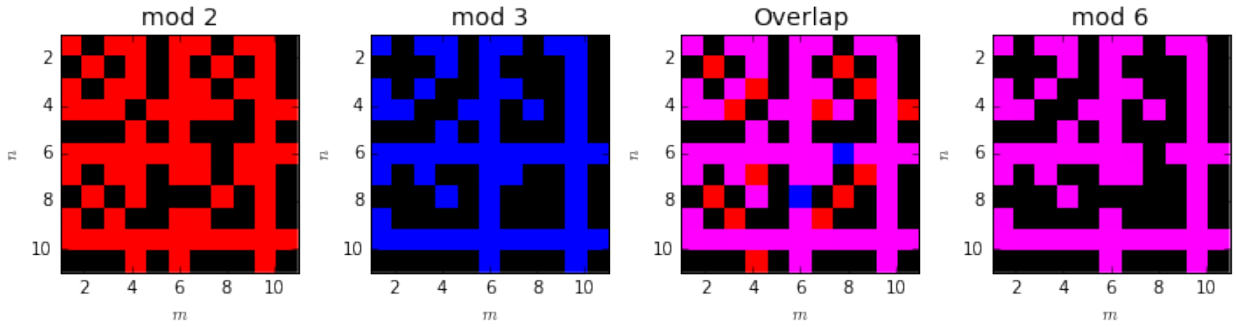


Figure 3: Comparison of full rank grid sizes for the standard transition matrix for mod 2, mod 3, and mod 6



of the form $Ax = b$ to be solved efficiently chaining 2 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 Euclids 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 a couple entries of 1 in the middle column.

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. In the previous example, full pivoting allows use to move the 1 in the center of the matrix to the active diagonal spot for factorization. Although full pivoting work 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, consider the transition matrix for the standard 5x5 lights out game. The matrix 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 a solution space of 2^23 states and null space of 2^2 states. We have adapted our algorithm to deal with the singular cases with multiple solutions for a prime modulus.

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

TODO

Applications of finding multiple solutions of a singular matrix include finding eigenvalues and quiet states.

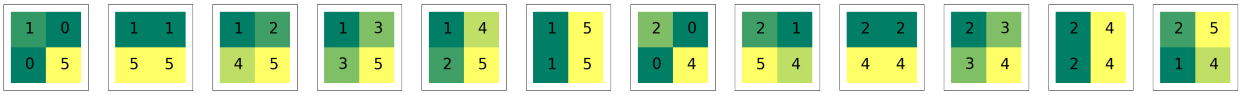
The 5x5 cases has a nullity of 2.

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.

Challenges with Composite Modulus

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

Figure 4: The twelve eigenstates of $A_{\%6}$ on a 2x2 grid



$3x = 0$ 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.

Eigengrids and Solution Cycles

Eigengrids (with eigenvalue 1) are characterized by the following equation:

$$Ax = x, \tag{2}$$

and repeating solution "cycles" (of period n) are characterized by:

$$A^n x = x, \tag{3}$$

or in other words, x is an eigengrid of A^n. For certain dimensions, modulos, and transition functions, we found that A^n commonly tended to equal the identity matrix for some n. We also found cases where

$$\begin{aligned} A^{n_1+n_2}x &= A^{n_1}x, \\ A^n x &\neq x \text{ for all } n, \end{aligned} \tag{4}$$

that is, where after n_1 "iterations" of solving solutions, we would enter a repeating solution cycle with period n_2 that didn't contain our initial set of n_1 transient states.

Let us consider the example of our normal Lights Out transition matrix $A\%6$ on a 2x2 grid. We choose this example in part because there are a manageable but large number of possible grids ($6^{2\times 2} = 1296$), and in part because our calculated nullity for this matrix disagrees with the predictions of [3]. We will iterate through each of the 1296 possible button grids b, solve $Ax = b$, and repeatedly re-solve our solutions until we reach a steady state.

First, we find that we appear to be able to solve $Ax = b$ for every b, despite the fact that $\det(A) = -3$ is not coprime to $k = 6$. Second, we find that for this case (unlike the set of solutions at lower values of k, which we also calculated), there are many transient states, that, when repeatedly solved, lead to a cycle that does not contain them.

	Sq. Grid Length				
k	**1**	**2**	**3**	**4**	**5**
2	2	4	8	∅	∅
3	3	∅	27	∅	∅
4	4	16			
5	5	25			

Figure 5: A 3-state transient path ending in a 6-state cycle of $A_{\%6}$ on a 2x2 grid

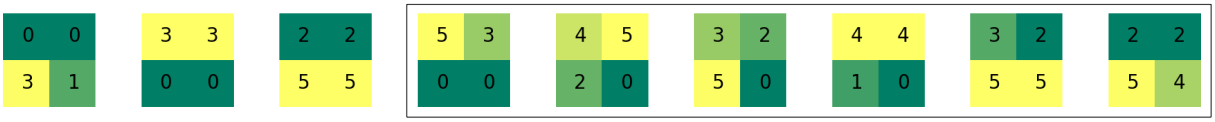
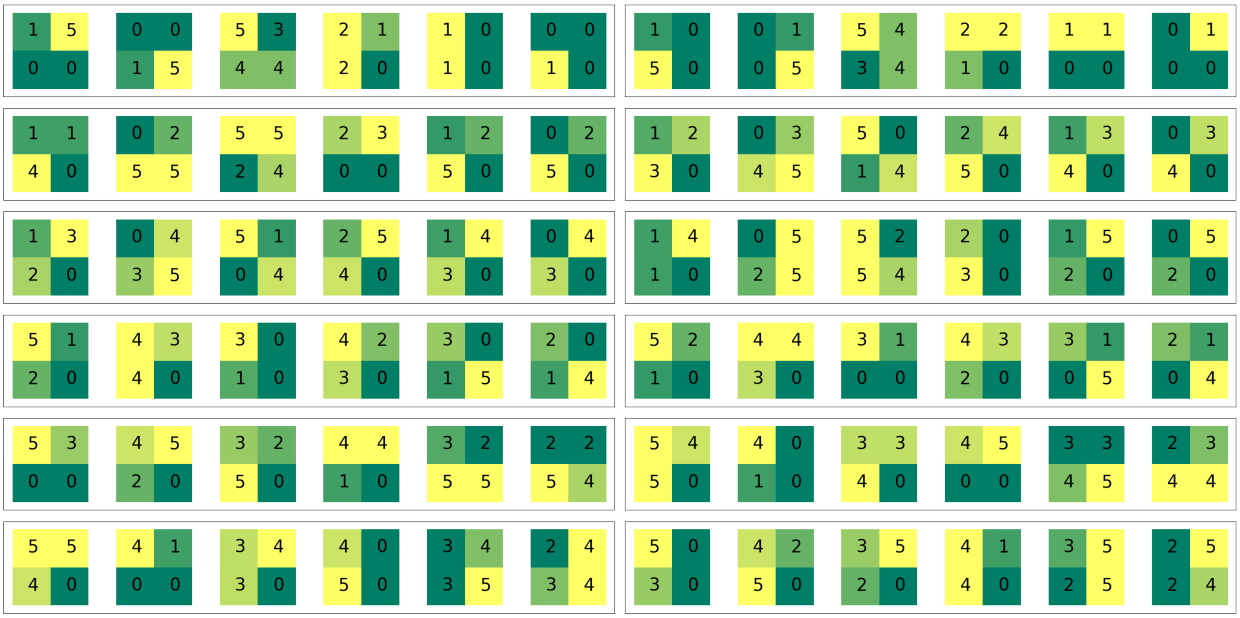


Figure 6: The twelve 6-state cycles of $A_{\%6}$ on a 2x2 grid



Visually Appealing Solutions

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.pdflink.

[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.