A Confused Electrician Uses Smith Normal Form

Tom Edgar
Pacific Lutheran University
Tacoma, WA 98447-0003
edgartj@plu.edu

Jessica K. Sklar
Pacific Lutheran University
Tacoma, WA 98447-0003
sklarjk@plu.edu

Several years ago, Jessica moved into a quirky home. Among its mysteries were a set of floodlights she was unable to turn on and a closet lamp that seemed to have a surplus of switches: both a light toggle in the closet and a chain hanging from the lamp turned the light on and off. Eventually, Jessica called an electrician. After serious study, he solved the puzzle: The light toggle switched the states of both the closet lamp AND the floodlights between on and off. So if the floodlights were on and the closet lamp were off, to turn them all off the electrician would have to both flip the light switch (turning the floodlights off and the closet lamp on), and then yank the chain (turning the closet lamp off).

This scenario provides a simple example of what we will call a "Confused Electrician game." Imagine that an electrician happens upon a collection of n lamps. For each $i = 1, 2 \ldots, n$, lamp i has $L_i > 1$ levels of brightness, ranging from brightness level 0, corresponding to the lamp being off, to brightness level $L_i - 1$, corresponding to the lamp being in its brightest state. The lamps are initially set to varied brightness levels (we call this configuration the initial state), and the electrician would like to simultaneously change the brightness levels to achieve another (usually different) brightness configuration (called the final state). Now, each lamp comes equipped with a button. However, there's a catch: pressing the button on lamp j (henceforth referred to as *button* j) adjusts the brightness of a subcollection of the n lamps, which may or may not contain lamp j itself. Specifically, pressing button j increments the brightness level of lamp i ($i = 1, 2, \ldots, n$) by a fixed nonnegative integer $b_{i,j}$, where we increment modulo L_i: when the lamp is at its brightest level, $L_i - 1$, "increasing" this level by 1 actually turns the lamp off.

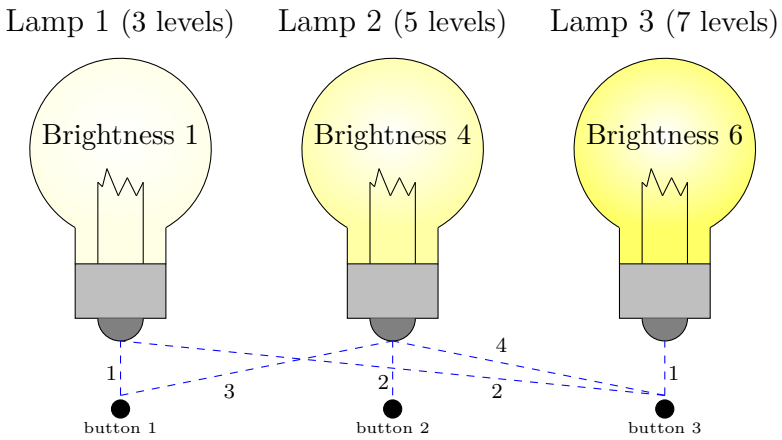


Figure 1: An example of a Confused Electrician game.

Since the wiring between lamps and buttons is so bizarre, the electrician is confused about how to achieve the desired final state; can she do it, and if so what buttons should she press? For instance, Figure 1 contains a diagram depicting a Confused Electrician game with three lamps. Lamp 1 has three brightness levels, lamp 2 has five brightness levels, and lamp 3 has seven brightness levels. Each button is connected to some lamps via wires (drawn as blue dashed lines), and pressing a button increments the brightness levels of the connected lamps by the integer labeling the wire. For instance, pressing button 1 increments the brightness of lamp 1 by one and lamp 2 by three, while not affecting lamp 3. In the beginning, lamps 1–3 are at brightness levels 1, 4, and 6, respectively. Can the electrician turn off all of the lamps using these buttons? We leave this as an exercise for the reader. (A solution is provided in an appendix at the end of this paper.)

Many problems and games appearing in mathematics literature can be interpreted as Confused Electrician games. Perhaps the most famous of these is "Lights Out," an electronic game released by Tiger Toys in 1995, which was described mathematically in [1]. A precursor of Lights Out, called "Magic Square,"[1] was a collection of games programmed into *Merlin, the Electronic Wizard,* an electronic game released by Parker Brothers in 1978 (see [7], [10], [12], and [16]). And a variety of more recently produced computer games,

[1]Merlin's Magic Square games should not be confused with the "magic squares," which consist of arrangements of numbers in square grids, where the numbers in each of the grid's rows, columns, and main diagonals sums to a common number.

including "Myst," "The Longest Journey," "Timelapse," and "Call of Duty: Black Ops," include cleverly disguised examples of these games (see [13] and [9]). We join the ranks of those studying Lights Out and related games (see, for example, [17], [8], [11], [4], [2], and [5]), providing a universal method for determining the existence of solutions to Confused Electrician games.

Modeling Confused Electrician games

Throughout, let $\mathbb{Z}$ be the set of integers, $\mathbb{Z}^+$ the set $\{1, 2, \ldots\}$ of positive integers, and $\mathbb{N}$ the set $\{0, 1, \ldots\}$ of natural numbers. For each $n \in \mathbb{Z}^+$, we denote by $\mathbb{Z}^n$ the group

$$\underbrace{\mathbb{Z} \oplus \cdots \oplus \mathbb{Z}}_{n \text{ copies}}.$$

Finally, for each $n \in \mathbb{Z}^+$, we let $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$, noting that $\mathbb{Z}_n$ is a group under addition modulo n.

We now review some graph theory concepts. (See, for instance, [18] or [19] for a more thorough discussion of graph theory.) Recall that a *directed graph* or *digraph* $\Gamma = (V, E)$ consists of a set V of *vertices* and set $E \subseteq V \times V$ of *(directed) edges*.[2] A digraph is *finite* if V is finite, and a digraph Γ is $\mathbb{Z}^+$-*weighted* if we assign to each edge in Γ a unique positive integer, called the edge's *weight*. We will draw weighted digraphs by letting integers represent vertices and integer-labeled arrows represent edges. If, for two vertices v and w, there is an edge from v to w and an edge from w to v that share a common weight, we instead draw a single bi-directed arrow between v and w.

Digraphs play a central role in Confused Electrician games; in fact, we will call a finite $\mathbb{Z}^+$-weighted digraph a *Confused Electrician graph*, or *CE-graph*. In turn, a *Confused Electrician game*, or *CE-game*, is a tuple

$$(\Gamma, \{G_i\}_{i=1}^n, \boldsymbol{s}, \boldsymbol{f}),$$

where

- Γ is a CE-graph with vertex set $\{1, 2, \ldots, n\}$;
- Each G_i is a finite cyclic group; and
- $\boldsymbol{s}, \boldsymbol{f} \in \bigoplus_{i=1}^n G_i$.

[2]Notice that our definition of digraphs allows a loop from a vertex to itself, but does not allow multiple edges from one vertex to another.

Given such a game, we call each G_i a *vertex group*, and $G := \bigoplus_{i=1}^{n} G_i$ the *board space* of the game. Note that G is a group under the operation that is determined, coordinatewise, by the operations in the G_i; since G is abelian, we will denote its operation by $+$. We call the elements of G *board states*; $\boldsymbol{s}$ and $\boldsymbol{f}$ are, respectively, the game's *initial* and *final* states.

From a confused electrician's perspective, n is the number of lamps, and each $G_i = \mathbb{Z}_{L_i}$, where L_i is the number of brightness levels of lamp i. The lamps initially have the brightness levels indicated by $\boldsymbol{s}$, and the electrician wishes to obtain the brightness levels indicated by $\boldsymbol{f}$. The edge weights in Γ encode the effects of pressing buttons. In particular, if the edge from vertex j to vertex i has weight $b_{i,j}$, then pressing button j increments the brightness of lamp i by $b_{i,j}$, modulo L_i.

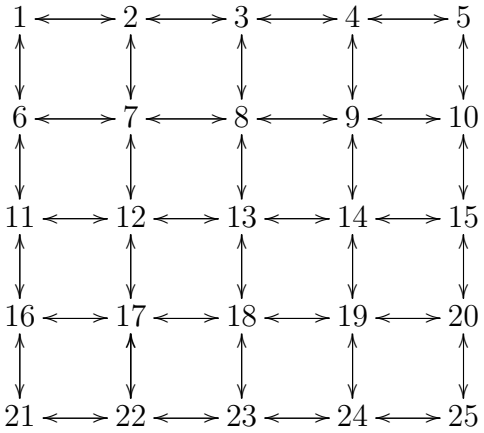
Examples of CE-games

We now provide just a few examples of the many puzzles that can be interpreted as CE-games. The graphs associated with the following games are provided in Figure 2.

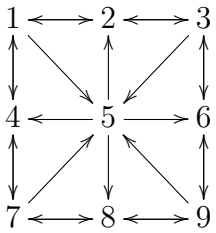
1. Lights Out is a collection of CE-games where Γ is the 5×5 lattice with loops added at each vertex and with each edge having weight 1. Moreover, each vertex group is $\mathbb{Z}_2$, so $G = \mathbb{Z}_2^{25}$. The initial states vary from game to game, but the final state is always $(0, 0, \ldots, 0)$. Note that this means that the goal of each game is to turn off all of the lamps.

2. Merlin's Magic Square is a collection of CE-games where Γ is as pictured in Figure 2, with loops added at each vertex and with each edge having weight 1. Again, each vertex group in a Magic Square game is $\mathbb{Z}_2$, and the games have varying initial states. However, the final state is always $(1, 1, 1, 1, 0, 1, 1, 1, 1)$; that is, the goal of the game is always to have only middle light off.

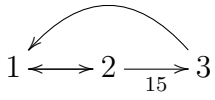
3. Several games described in [13] can be modeled by CE-games. In particular, Timelapse's Mayan Calendar puzzles can be interpreted as CE-games with three distinct vertex groups. The associated CE-graph Γ is as shown in Figure 2, with loops of weight 1 added at each vertex. In these games, $G_1 = \mathbb{Z}_8$, $G_2 = \mathbb{Z}_{12}$, and $G_3 = \mathbb{Z}_{16}$, while $\boldsymbol{s}$ and $\boldsymbol{f}$ vary from game to game.



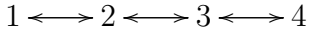
Lights Out



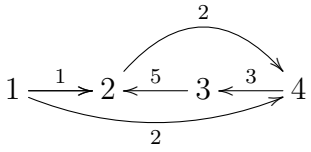
Merlin's Magic Square



Timelapse's Mayan Calendar



Call of Duty: Black Ops (#1)



Call of Duty: Black Ops (#2)

Figure 2: The CE-graphs for five CE-games. All loops have been omitted and all non-labeled have weight 1.

4. Two puzzles from Call of Duty: Black Ops that appear in [9] can be modeled as CE-games. Their graphs are as shown in Figure 2, with loops weighted by 1 added at each vertex. In both cases, each vertex group is $\mathbb{Z}_{10}$, $\boldsymbol{s} = (8, 8, 4, 5)$, and $\boldsymbol{f} = (2, 7, 4, 6)$.

Winning Confused Electrician games

So how does a confused electrician go about trying to win a CE-game? Recall that if the edge from vertex j to vertex i has weight $b_{i,j}$, then pressing button j increments the brightness of lamp i by $b_{i,j} \pmod{L_i}$. If there is no edge from vertex j to vertex i, we let $b_{i,j} = 0$. Then, for each $j = 1, 2, \ldots, n$, we define a vector

$$\boldsymbol{b}_j = (b_{1,j}, b_{2,j}, \ldots, b_{n,j}) \in \mathbb{Z}^n,$$

which we call the *button vector* of vertex j.[3] For example, in Lights Out, the button vector of vertex 7 is

$$\boldsymbol{b}_7 = (0,1,0,0,0,1,1,1,0,0,0,1,0,0,0,0,0,0,0,0,0,0,0,0,0) \in \mathbb{Z}^{25},$$

and in Merlin's Magic Square, the button vector of vertex 3 is

$$\boldsymbol{b}_3 = (0,1,1,0,1,1,0,0,0) \in \mathbb{Z}^9.$$

A CE-game can be won if and only if there exists a finite sequence of button vectors the electrician can push that will move the game into its final state. Since addition in G is commutative, if we start in a state and press button i and then button j, we obtain the same state we would have obtained by pressing the buttons in the opposite order. So performing any finite sequence of button presses corresponds to adding an $\mathbb{N}$-linear combination of the game's button vectors to the game's current state. Thus, letting π be the canonical projection from $\mathbb{Z}^n$ to G, we say a CE-game $(\Gamma, \{G_i\}_{i=1}^n, \boldsymbol{s}, \boldsymbol{f})$ is *winnable* if there exists an $\mathbb{N}$-linear combination $\boldsymbol{p}$ of button vectors such that

$$\boldsymbol{s} + \pi(\boldsymbol{p}) = \boldsymbol{f}.$$

In this case, we call $\boldsymbol{p}$ a *solution* for the game.

[3] While it is usually most useful for us to think of each $\boldsymbol{b}_j$ as a column vector, for the sake of appearances, we will often typeset such vectors as n-tuples, and not bother using transpose notation in those cases.

In order to demonstrate the ideas from the previous paragraph, consider the Magic Square game with initial state

$$\boldsymbol{s} = (0, 0, 1, 0, 0, 0, 0, 1, 1),$$

and desired final state

$$\boldsymbol{f} = (1, 1, 1, 1, 0, 1, 1, 1, 1).$$

This game can be won by pressing buttons 1, 8, and 9, since in $\mathbb{Z}_2^9$,

$$\boldsymbol{s} + \pi(\boldsymbol{b}_1 + \boldsymbol{b}_8 + \boldsymbol{b}_9) = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \end{bmatrix} + \pi \left(\begin{bmatrix} 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 1 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \end{bmatrix} \right) = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = \boldsymbol{f}.$$

We now build machinery we can use to win a CE-game $(\Gamma, \{G_i\}_{i=1}^n, \boldsymbol{s}, \boldsymbol{f})$. First, for simplicity, we define $\boldsymbol{\delta} \in G$ by $\boldsymbol{\delta} = \boldsymbol{f} - \boldsymbol{s}$. As indicated above, a CE-game is winnable if and only if there exists a collection of nonnegative integers $\{\lambda_i\}_{i=1}^n$, such that

$$\boldsymbol{s} + \pi(\lambda_1 \boldsymbol{b}_1 + \cdots + \lambda_n \boldsymbol{b}_n) = \boldsymbol{f},$$

that is, such that

$$\pi(\lambda_1 \boldsymbol{b}_1 + \cdots + \lambda_n \boldsymbol{b}_n) = \boldsymbol{\delta}.$$

We note that λ_i represents the number of times we must press button i in order to win the game.

Finding integer solutions to the equation

$$\pi(x_1 \boldsymbol{b}_1 + \cdots + x_n \boldsymbol{b}_n) = \boldsymbol{\delta}$$

is equivalent to finding integers that solve the simultaneous system of congruences

$$\begin{aligned} x_1 b_{1,1} + \cdots + x_n b_{1,n} &\equiv \delta_1 \pmod{L_1} \\ &\vdots \\ x_1 b_{n,1} + \cdots + x_n b_{n,n} &\equiv \delta_n \pmod{L_n}, \end{aligned} \tag{1}$$

where δ_i is the ith coordinate of $\boldsymbol{\delta}$.

Note that a solution to this mathematical system may contain negative integers, which would mean that it does not represent a viable solution for the game (as one cannot press a button a negative number of times). However, if M is the least common multiple of the brightness levels $L_1, L_2, \ldots, L_n$, and $\boldsymbol{M}$ the element of $\mathbb{Z}^n$ with each entry equal to M, then we see that an element $\boldsymbol{\lambda} \in \mathbb{Z}^n$ solves the system if and only if $\boldsymbol{\lambda} + \boldsymbol{M}$ solves the system; consequently, a solution with negative entries can be replaced by a solution with all nonnegative entries by adding an appropriate integer multiple of $\boldsymbol{M}$. Thus, we have the following theorem:

Theorem 1. *A CE-game is winnable if and only if its corresponding system of congruences, as described in (1), has a solution in $\mathbb{Z}^n$.*

The optimal technique used to solve a game's system varies from game to game. In most of the games we've mentioned, each vertex group is the same; indeed in the Lights Out and Magic Square games each G_i is $\mathbb{Z}_2$; in the Myst puzzle and in one of the Timelapse puzzles discussed in [13], each $G_i = \mathbb{Z}_3$; and in Call of Duty: Black Ops, each $G_i = \mathbb{Z}_{10}$. If each congruence involves the same prime modulus, we can use basic linear algebraic techniques to solve the system. Even if the congruences utilize the same composite modulus, standard matrix methods may still suffice (see [9]).

When the vertex groups vary, as they do in the Timelapse Mayan Calendar games, we may have to do a little more work. The systems of congruences corresponding to the Mayan Calendar games are simple enough that they are relatively easily solved via brute force, but we will see later that this method may be impractical for solving an arbitrary CE-game when Γ contains many vertices or when the L_i are large.

A sufficient condition for winning a CE-game

The following lemma, whose straightforward proof is left to the reader, provides a method that will allow us to solve some systems of linear congruences involving differing moduli.

Lemma 2. *Let $\{\lambda_i\}_{i=1}^n$ and $\{a_i\}_{i=1}^n$ be collections of n integers, and let v be an integer. Then for any integer multiple M of L,*

$$\lambda_1 a_1 + \lambda_2 a_2 + \cdots + \lambda_n a_n \equiv v \pmod{L}$$

if

$$\lambda_1 a_1 + \lambda_2 a_2 + \cdots + \lambda_n a_n \equiv v \pmod{M}.$$

Applying this lemma with $M := \text{lcm}(L_1, \ldots, L_n)$, we conclude that the system of congruences described in (1) will have a solution if the system

$$\begin{aligned} x_1 b_{1,1} + \cdots + x_n b_{1,n} &\equiv \delta_1 \pmod{M} \\ &\vdots \\ x_1 b_{n,1} + \cdots + x_n b_{n,n} &\equiv \delta_n \pmod{M} \end{aligned} \tag{2}$$

has a solution. We call the latter system of congruences the *uniform-group system* of the game.

Notice that the lemma only provides a *sufficient* condition for existence of a solution to the original system. The converse of the lemma does not hold; even if the modified system does not have a solution, the original system may still have a solution. We provide an example of this below.

Now, we can use a computer-algebra system to solve a CE-game assuming its uniform-group system of congruences has a solution. In particular, the open-source software Sage ([14]) provides a command "`solve_mod`" that will solve such systems of congruences. For instance, revisiting our Merlin Magic Square example from the previous section, we have

$$\begin{aligned} \boldsymbol{\delta} &= \boldsymbol{f} - \boldsymbol{s} \\ &= (1,1,1,1,0,1,1,1,1) - (0,0,1,0,0,0,0,1,1) \\ &= (1,1,0,1,0,1,1,0,0). \end{aligned}$$

We can read off $\boldsymbol{b}_1, \ldots, \boldsymbol{b}_9$ from the Magic Square digraph (Figure 2). In this case, each $L_i = 2$, so we use $M = 2$. We then use Sage's `solve_mod` command, with modulus 2, to obtain the result $(1,0,0,0,0,0,0,1,1)$. Thus, one solution to the game is given by letting $x_1 = x_8 = x_9 = 1$, and letting every other variable equal 0. (Note: We could also have solved this system by inverting the matrix $\begin{bmatrix} \boldsymbol{b}_1 & \boldsymbol{b}_2 & \cdots & \boldsymbol{b}_n \end{bmatrix}$ over $\mathbb{Z}_2$ and multiplying by $\boldsymbol{\delta}$.)

Unfortunately, the previous technique doesn't always work. For example, consider the Timelapse Mayan Calendar game with $\boldsymbol{s} = (4,10,12)$ and $\boldsymbol{f} = (0,0,0)$, so that

$$\boldsymbol{\delta} = (0,0,0) - (4,10,12) = (4,2,4) \in \mathbb{Z}_8 \oplus \mathbb{Z}_{12} \oplus \mathbb{Z}_{16}.$$

In this game, we have $\boldsymbol{b}_1 = (1, 1, 0)$, $\boldsymbol{b}_2 = (1, 1, 15)$, and $\boldsymbol{b}_3 = (1, 0, 1)$, giving us the following system of congruences to solve:

$$\begin{aligned} x_1 + x_2 + x_3 &\equiv 4 \pmod 8 \\ x_1 + x_2 &\equiv 2 \pmod{12} \\ 15x_2 + x_3 &\equiv 4 \pmod{16}. \end{aligned}$$

In this game, $M = \operatorname{lcm}(8, 12, 16) = 48$. As in the previous example, we use Sage's `solve_mod` command, but this time it provides no solutions. Does this mean that the game is not winnable? *Not necessarily.* Indeed, there is no solution to the uniform-group system; however, one can confirm that $x_1 = 12$, $x_2 = 2$, and $x_3 = 38$ is a solution to the original system. Unfortunately, Lemma 2 does not provide us with this solution.

A necessary condition for winning a CE-game

In order to find a solution for the game in the previous example, we determine a necessary condition for the existence of a solution to a system of equations with differing moduli. Suppose we have a fixed CE game $(\Gamma, \{G_i\}_{i=1}^n, \boldsymbol{s}, \boldsymbol{f})$. If we let $M = \operatorname{lcm}(L_1, \ldots, L_n)$, then we have $2 \leq L_i \leq M$ for each i. Thus, we can interpret each L_i as being an element of the finite group $\mathbb{Z}_M$, where if $L_i = M$ in $\mathbb{Z}$ we of course interpret it as the element 0 in $\mathbb{Z}_M$. We let $\langle L_i \rangle$ represent the cyclic subgroup of $\mathbb{Z}_M$ generated by L_i. For example, when $L_1 = 2$, $L_2 = 3$, and $L_3 = 4$, we have $M = \operatorname{lcm}(2, 3, 4) = 12$, and, for instance, $\langle L_1 \rangle = \langle 2 \rangle = \{0, 2, 4, 6, 8, 10\}$ in $\mathbb{Z}_{12}$. Finally, we note that since $0 \leq \delta_i < L_i$ for each i, we can also interpret each δ_i as an element of $\mathbb{Z}_M$.

This terminology allows us to improve Lemma 2.

Theorem 3. *Let $\{\lambda_i\}_{i=1}^n$ and $\{a_i\}_{i=1}^n$ be collections of n integers, and let v be an integer. Then for any integer multiple M of L,*

$$\lambda_1 a_1 + \lambda_2 a_2 + \cdots + \lambda_n a_n \equiv v \pmod L$$

if and only if

$$\lambda_1 a_1 + \lambda_2 a_2 + \cdots + \lambda_n a_n \equiv w \pmod M$$

for some $w \in v + \langle L \rangle$ in $\mathbb{Z}_M$.

Note that the order of $\langle L \rangle$ in $\mathbb{Z}_M$ is M/L, so there are only finitely many possibilities for w. Indeed, each w equals $v + kL$ for some $0 \leq k < M/L$.

This theorem allows us to find a solution for our Mayan Calendar game with $\boldsymbol{s} = (4, 10, 12)$ and $\boldsymbol{f} = (0, 0, 0)$. Recall that we must solve the system

$$\begin{aligned} x_1 + x_2 + x_3 &\equiv 4 \pmod 8 \\ x_1 + x_2 &\equiv 2 \pmod{12} \\ 15x_2 + x_3 &\equiv 4 \pmod{16}. \end{aligned}$$

By Theorem 3, this is equivalent to solving some system of the form

$$\begin{aligned} x_1 + x_2 + x_3 &\equiv 4 + 8k_1 \pmod{48} \\ x_1 + x_2 &\equiv 2 + 12k_2 \pmod{48} \\ 15x_2 + x_3 &\equiv 4 + 16k_3 \pmod{48}, \end{aligned}$$

where $k_1, k_2, k_3 \in \mathbb{Z}$ with $0 \leq k_1 < 6$, $0 \leq k_2 < 4$, and $0 \leq k_3 < 3$. Using Sage to solve all 72 such systems, we obtain, for instance, the solution $(12, 2, 38)$, when $k_1 = 0$, $k_2 = 1$, and $k_3 = 1$. There may be other solutions as well.

Classifying winnable CE-games using Smith normal form

In the previous section, we demonstrated that we can numerically solve a CE-game using a computer algebra system such as Sage. However, our method relies on using the Sage "black box" command `solve_mod`: the actual mathematics used to solve a uniform-group system remains "hidden" in Sage's internal programming. Moreover, the following example shows that using Sage is inefficient when either the vertex groups or n are relatively large.

Suppose, for instance, that we use the Petersen graph (Figure 3) as the CE-graph of a CE-game where each $G_i = \mathbb{Z}_{20}$, $\boldsymbol{s} = (1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$, and $\boldsymbol{f} = (0, 0, 0, 0, 0, 0, 0, 0, 0, 0)$. (Note that in this graph, there are no loops at vertices; the edges in the graph are exactly those pictured in the figure.) It takes the built-in feature in Sage approximately 21 *minutes* to solve the corresponding system of inequalities (modulo 20)!

So let's discuss a method to *efficiently* solve arbitrary CE-games. Given a CE-game with button vectors $\boldsymbol{b}_1, \boldsymbol{b}_2, \ldots, \boldsymbol{b}_n$, we let B be the $n \times n$ matrix with jth column $\boldsymbol{b}_j$; we call this the *button matrix* of the game. We note that the button matrix of a CE-game is an integer matrix; specifically, it is the weighted adjacency matrix of the corresponding CE-graph.

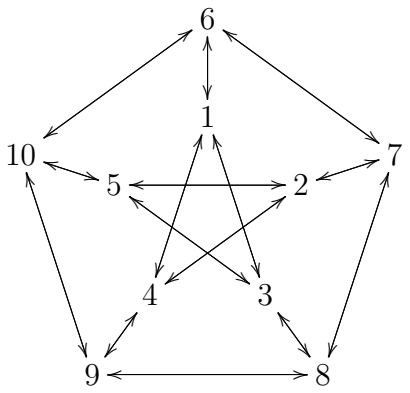


Figure 3: The Petersen graph.

For the remainder of this section, we fix a CE-game $(\Gamma, \{G_i\}_{i=1}^n, \boldsymbol{s}, \boldsymbol{f})$, with each $G_i = \mathbb{Z}_{L_i}$ and with button matrix B. We again let M be the least common multiple of the L_i.

Since $\mathbb{Z}^n$ and $\mathbb{Z}_M^n$ are both $\mathbb{Z}$-modules, they are also modules over the ring of $n \times n$ integer matrices. For each $\boldsymbol{y} \in \mathbb{Z}^n$, $B\boldsymbol{y}$ is also an element of $\mathbb{Z}^n$, and so $\pi(B\boldsymbol{y}) \in G$. Similarly, for each $\boldsymbol{z} \in \mathbb{Z}_M^n$, $B\boldsymbol{z} \in \mathbb{Z}_M^n$. Using this language, the following result follows immediately from Theorem 1.

Lemma 4. *The CE-game with button matrix B is winnable if and only if the equation $\pi(B\boldsymbol{x}) = \boldsymbol{\delta}$ has a solution in $\mathbb{Z}^n$.*

Since M may not be prime or B may not be invertible over $\mathbb{Z}_M$, in order to find solutions of this new equation we will write B in a particular form. Any integer matrix C can be written in the form $C = UDV$, where U, D, and V are integer matrices satisfying the following two properties:

- Both U and V are invertible over $\mathbb{Z}$ (that is, each has determinant ± 1).

- The matrix D is diagonal of the form

$$\operatorname{diag}(a_1, a_2, \ldots, a_k, 0, 0, \ldots, 0) := \begin{bmatrix} a_1 & 0 & 0 & & \cdots & & 0 \\ 0 & a_2 & 0 & & \cdots & & 0 \\ 0 & 0 & \ddots & & & & 0 \\ \vdots & & & a_k & & & \vdots \\ & & & & 0 & & \\ & & & & & \ddots & \\ 0 & & & \cdots & & & 0 \end{bmatrix},$$

  where a_i evenly divides a_{i+1}, for each i.

This form for C is known as its *Smith normal form*, and the numbers $a_1, \ldots, a_k$ are known as the *elementary divisors* of C. (The process for computing U, V, and D is relatively straightforward, but outside of the scope of this paper; for more information see [3].)

Suppose that UDV is the Smith normal form for integral matrix B. For each $i \in \{1, 2, \ldots, n\}$, we let $\boldsymbol{e}_i$ be the vector in $\mathbb{Z}_M^n$ with ith coordinate equal to 1 and every other coordinate equal to 0. We next let S be the subgroup of $\mathbb{Z}_M^n$ generated by the set $\{L_1\boldsymbol{e}_1, L_2\boldsymbol{e}_2, \ldots, L_n\boldsymbol{e}_n\}$, and let A be the subgroup of $\mathbb{Z}_M^n$ generated by the set $\{a_1\boldsymbol{e}_1, a_2\boldsymbol{e}_2, \ldots, a_k\boldsymbol{e}_k\}$, where the a_i's are the elementary divisors of B. Recall that we can interpret $\boldsymbol{\delta}$ as an element of $\mathbb{Z}_M^n$ so that $U^{-1}\boldsymbol{\delta} + U^{-1}(S)$ is a subset of $\mathbb{Z}_M^n$. Then we have the following.

Theorem 5. *The equation $\pi(B\boldsymbol{x}) = \boldsymbol{\delta}$ has a solution in $\mathbb{Z}^n$ if and only if $A \cap (U^{-1}\boldsymbol{\delta} + U^{-1}(S)) \neq \emptyset$.*

Indeed, suppose there exists $\boldsymbol{w} \in A \cap (U^{-1}\boldsymbol{\delta} + U^{-1}(S))$: that is, assume there is an element $\boldsymbol{w}$ in $U^{-1}\boldsymbol{\delta} + U^{-1}(S)$ of the form

$$\boldsymbol{w} = (\lambda_1 a_1, \lambda_2 a_2, \ldots, \lambda_k a_k, 0, \ldots, 0),$$

where each $\lambda_i \in \mathbb{Z}$. We let $\boldsymbol{\lambda} = (\lambda_1, \lambda_2, \ldots, \lambda_k, 0, \ldots, 0) \in \mathbb{Z}^n$. Now in $\mathbb{Z}^n$, each coordinate of $D\boldsymbol{\lambda}$ is congruent, modulo M, to the corresponding coordinate of $\boldsymbol{w}$. Let

$$\boldsymbol{y} := V^{-1}\boldsymbol{\lambda} \in \mathbb{Z}^n.$$

Then in $\mathbb{Z}^n$,

$$B\boldsymbol{y} = (UDV)\boldsymbol{y} = UD(V\boldsymbol{y}) = UD\boldsymbol{\lambda},$$

and thus $B\boldsymbol{y}$ is congruent, coordinatewise modulo M, to $U\boldsymbol{w}$. Finally, $U\boldsymbol{w} \in \boldsymbol{\delta} + S$, which implies that the ith coordinate of $U\boldsymbol{w}$ is congruent, modulo L_i, to δ_i. Thus, the ith coordinate of $B\boldsymbol{y}$ is congruent, modulo L_i, to δ_i, and so we have $\pi(B\boldsymbol{y}) = \boldsymbol{\delta}$.

A similar argument, along with Theorem 3, yields that if $\boldsymbol{y} \in \mathbb{Z}^n$ solves the equation $\pi(B\boldsymbol{x}) = \boldsymbol{\delta}$, then $A \cap (U^{-1}\boldsymbol{\delta} + U^{-1}(S)) \neq \emptyset$.

Once we have $\boldsymbol{y} \in \mathbb{Z}^n$ satisfying $\pi(B\boldsymbol{y}) = \boldsymbol{\delta}$, we see that every element of the form $\boldsymbol{y} + V^{-1}\boldsymbol{v} \in \mathbb{Z}^n$, where $D\boldsymbol{v} = 0$, also solves the equation $\pi(B\boldsymbol{x}) = \boldsymbol{\delta}$. We can of course then obtain a solution of the CE-game—that is, an element in $\mathbb{N}^n$ solving $\pi(B\boldsymbol{x}) = \boldsymbol{\delta}$—by adding an appropriate integer multiple of $\boldsymbol{M}$ to any of those solutions.

We can use Sage to quickly produce the Smith normal form, UDV, of the button matrix for the Petersen graph in order to find a solution to the CE-game described at the beginning of this section. In this case, $S = \{(0,0,0,0,0,0,0,0,0,0)\}$ and $D = \mathrm{diag}(1,1,1,1,1,1,2,2,2,6)$, so we only need to check that the ith coordinate of $U^{-1}\boldsymbol{\delta}$ is in the cyclic subgroup of $\mathbb{Z}_{20}$ that is generated by a_i; in particular, we only need to check the last four coordinates of $U^{-1}\boldsymbol{\delta}$, since 1 is a generator of $\mathbb{Z}_{20}$. Then, using the ideas presented in this section, a simple for-loop will find a solution to the game (if one exists). In particular, one solution for this game is

$$(13,13,13,13,13,13,13,13,13,13).$$

For this example, the entire process in Sage takes approximately 0.01 *seconds*.

For CE-games with non-trivial S, a little extra programming is required in order to find an element in $A \cap (U^{-1}\boldsymbol{\delta} \cap U^{-1}(S))$ (or determine that no such element exists). Unfortunately, as S gets larger, the computation time gets longer using this method or the brute-force method. If S gets too large, both computational methods may be impractical, but using the Smith normal form will still generally be more efficient than solving the associated system of congruences.

Thus, we see that when a CE-game involves many lamps, or its lamps have many brightness levels, the brute force method of checking all possible solutions to the game can be wildly inefficient, while one can quickly solve the game using the Smith normal form of the button matrix. Finally, the result in Theorem 5 also allows us to construct initial and final states for a given CE-graph that will yield a winnable CE-game.

References

[1] Marlow Anderson and Todd Feil, Turning Lights Out with linear algebra, *Mathematics Magazine* **71**(5) (1998) 300–303.

[2] Crista Arangala, Maggie MacDonald, and Rachel Wilson, Multistate Lights Out, *Pi Mu Epsilon J.***14**(1) (2014) 9–18.

[3] David S. Dummit and Richard M. Foote, *Abstract Algebra* (3rd ed.), Wiley, 2003.

[4] Stephanie Edwards, Victoria Elandt, Nicholas James, Kathryn Johnson, Zachary Mitchell, and Darin Stephenson, Lights Out on finite graphs, *Involve* **3**(1) (2010), 17–32.

[5] Rudolf Fleischer and Jiajin Yu. A survey of the game "Lights Out!". In *Space-efficient data structures, streams, and algorithms*, volume 8066 of *Lecture Notes in Comput. Sci.*, pages 176–198. Springer, Heidelberg, 2013.

[6] John B. Fraleigh, *A First Course in Abstract Algebra (7th ed.)*, Addison Wesley, 2002.

[7] Richard A. Gibbs, Merlin^{TM} and the Magic Square, *The Mathematics Teacher* **75**(1) (1982) 78–81.

[8] Alexander Giffen and Darren B. Parker, On generalizing the "Lights Out" game and a generalization of parity domination, *Ars Combinatoria* **111** (2013) 273–288.

[9] Heidi Hulsizer, A 'Mod'ern mathematical adventure in "Call of Duty: Black Ops", *Math Horizons* **21**(3) (2014) 12–15.

[10] Jennie Missigman and Richard Weida, An easy solution to mini Lights Out, *Mathematics Magazine* **74**(1) (2001) 57–59.

[11] Torsten Muetze, Generalized switch-setting problems, *Discrete Mathematics* **307**(22) (2007), 2755–2770.

[12] Don Pelletier, Merlin's Magic Square, *The American Mathematical Monthly* **94**(2) (1987) 143–150.

[13] Jessica Sklar, Dials and levers and glyphs, oh my! Linear algebra solutions to computer game puzzles, *Mathematics Magazine* **79**(5) (2006) 360–367.

[14] William A. Stein et al., *Sage Mathematics Software (Version 6.2)*, The Sage Development Team, 2014, `http://www.sagemath.org`.

[15] William A. Stein et al., *Sage's Version 6.2 Reference Manual*, The Sage Development Team, 2014, `http://www.sagemath.org/doc/reference`.

[16] Daniel L. Stock, Merlin's Magic Square revisited, *The American Mathematical Monthly* **96**(7) (1989) 608–610.

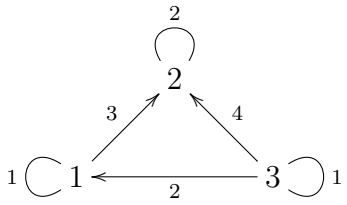
[17] Bruce Torrence and Robert Torrence, Lights Out on Petersen graphs, 2013,
`http://graphics8.nytimes.com/packages/blogs/images/LightsOutPetersen-Torrence2B.pdf`.

[18] Douglas B. West, *Introduction to Graph Theory (2nd ed.)*, Prentice-Hall, Inc., 2000.

[19] Robin J. Wilson and John J. Watkins, *Graphs: An Introductory Approach—A First Course in Discrete Mathematics*, Wiley, 1990.

Appendix

The CE-game pictured in Figure 1 has the following CE-graph, with $G_1 = \mathbb{Z}_3$, $G_2 = \mathbb{Z}_5$, $G_3 = \mathbb{Z}_7$, $\boldsymbol{s} = (1, 4, 6)$, and $\boldsymbol{f} = (0, 0, 0)$.



Thus, $\boldsymbol{\delta} = (2, 1, 1)$, and the button matrix is given by

$$B = \begin{bmatrix} 1 & 0 & 2 \\ 3 & 2 & 4 \\ 0 & 0 & 1 \end{bmatrix}.$$

Now, we can check that $\boldsymbol{y} = (0, 1, 1)$ is a solution since

$$\begin{bmatrix} 1 & 0 & 2 \\ 3 & 2 & 4 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 \\ 6 \\ 1 \end{bmatrix},$$

which is equivalent, coordinatewise, to $(2, 1, 1)$. Thus, $\pi(B\boldsymbol{y}) = \boldsymbol{\delta}$.