 doi:10.20944/preprints202206.0331.v1

*Article*

# Two-State Alien Tiles: A Coding-Theoretical Perspective

**Hoover H. F. Yin [1]*, Ka Hei Ng [2], Shi Kin Ma [3], Harry W. H. Wong [4] and Hugo Wai Leung Mak [5,6]***

[1] Institute of Network Coding, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong; hfyin@inc.cuhk.edu.hk
[2] Department of Physics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong; kaheicanaan@link.cuhk.edu.hk
[3] Department of Computer Science and Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong; km526829@gmail.com
[4] Department of Information Engineering, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong; wwh016@ie.cuhk.edu.hk
[5] Department of Mathematics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong; hwlmak@math.cuhk.edu.hk
[6] Department of Mathematics, The Hong Kong University of Science and Technology, Clear Water Bay, Kowloon, Hong Kong; hwlmak@ust.hk
* Correspondence: hfyin@inc.cuhk.edu.hk (H. H. F. Yin), hwlmak@ust.hk (H. W. L. Mak)

**Abstract:** The switching game Lights Out and its variants were studied extensively as recreational mathematics problems. The game board of the ordinary Lights Out is a rectangular grid of lights, where each light is either on or off. By clicking a light, the clicked light and its adjacent rectilinear neighbors are toggled. Given an arbitrary initial configuration of lights, the final mission is to "solve" this game by switching off all the lights. Most studies on Lights Out and its variants focused on the solvability of given games or the number of solvable games, but when the game is viewed in a coding-theoretical perspective, more interesting questions with special meanings in coding theory will naturally pop up, such as finding the minimal number of lit lights among all solvable games except the solved game, or finding the minimal number of lit lights that the player can achieve from a given unsolvable game, etc. However, these problems are usually hard to be solved in general in terms of algorithmic complexity. This study considers a natural extension of the Lights Out game, which enlarges the toggle pattern in a way that all the lights in the same row and those in the same column of the clicked light are toggled. This variant of Lights Out is a two-state version of a switching game called Alien Tiles. In this paper, we investigate the properties of the two-state Alien Tiles, and discuss several coding-theoretical problems about this game. Then, we apply this game as an error correction code and investigate its optimality. We also give a brief overview on algorithmic complexity and coding theory for readers who are not familiar with these topics. The purpose of this paper is to propose ways of playing switching games in a think-outside-the-box manner, which benefit the recreational mathematics community.

**Keywords:** Alien Tiles; Coding Theory; Lights Out; Recreational Mathematics; Abstract Algebra

## 1. Introduction

The mathematics of *Lights Out* was extensively studied in the past decades. The Lights Out game board is a rectangular grid of lights, where each light is either on (lit) or off (unlit). By performing a *move*, a selected light and its adjacent rectilinear neighbors are toggled. At the beginning of the game, some lights have already been switched on. The player is asked to *solve* the game, i.e., switching off all the lights, by performing a sequence of moves. Many variants such as the setting up of different toggle patterns were explored, e.g., in [1,2].

Regardless of which variant of Lights Out, most studies focused on the solvability (or the attainability) of given games, or the number of solvable (or the attainable) games.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



These problems can usually be solved efficiently by elementary linear algebra and group theory approaches, e.g., in [3,4]. More precisely, these problems are in the complexity class P. A brief introduction to algorithmic complexity can be found in Section A. In a nutshell, problems in P can be computed efficiently, and are considered as simple mathematical problems. Sometimes, the analysis may require the use of other mathematics tools, such as Fibonacci polynomials [5] and cellular automata [6].

However, not every derived problem has a known algorithm to be solved efficiently, for example, the *shortest solution problem* focuses on solving a game with minimal number of moves, which is related to the minimum distance decoding (or the maximum likelihood decoding) in coding theory. In a more general setting, this problem was proven to have no known efficient algorithms for solving [7,8]. Precisely, this problem is NP-hard and its decision problem is NP-complete. Approximating the solution within a constant factor is also NP-hard [9,10]. In layman terms, NP-complete problems are considered as hard problems where the existence or the non-existence of efficient ways to compute or approximate the solution is still an open problem. NP-hard problems are at least as hard as NP-complete problems, thus they are also considered as hard problems. Yet, it is possible to solve the shortest solution problem efficiently if certain special properties of the game can be detected and found.

The motivation of this study is inspired by the *Gale-Berlekamp switching game*, a Lights Out variant where the player can either toggle an entire row or an entire column per move. Example moves of a $5 \times 5$ Gale-Berlekamp switching game are illustrated in Fig. 1, where the gray cells indicate the toggle pattern. Despite the simple toggle patterns, Roth and Viswanathan [11] proved that finding the minimal number of lights that the player cannot switch off from a given Gale-Berlekamp switching game is NP-complete, although a linear time approximation was later discovered by Karpinski and Schudy [12]. This problem has a special meaning in coding theory, and has attracted researchers to further investigate this game, e.g., [13–15]. In other words, when we view the game in a coding-theoretical perspective, we can ask more questions with special meanings in coding theory, such as:

- **Hamming weight**. What is the minimal number of lit lights among all solvable games except the solved game?
- **Coset leader**. Which game has the minimal number of lit lights that the player can achieve from a given game? What is this minimal number? How many such games can the player achieve?
- **Covering radius**. Among all possible games, what is the maximal number of lit lights that remained when the number of lit lights is minimized?
- **Error correction**. Which is the "closest" solvable game from a given unsolvable game in the sense of toggling the minimal number of individual lights? Further, is such closest solvable game unique?

A brief introduction to coding theory can be found in Section B, which we discuss the physical meaning of the aforementioned terminologies in coding theory. Again, not all problems have known algorithms to be solved in an efficient manner. For example, the first problem about Hamming weight, which is also known as the *minimum distance problem*, is NP-hard in general [16].

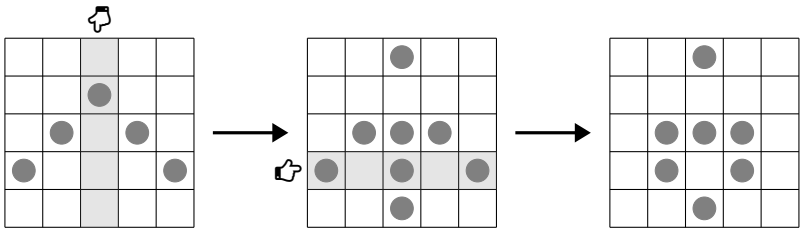


**Figure 1.** Example moves of a $5 \times 5$ Gale-Berlekamp switching game.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



In this study, we consider a natural variant of Lights Out in a way that each toggle pattern is changed from a "small cross" to a "big cross" that covers the entire row and the entire column. This variant is the two-state version of a game called *Alien Tiles*, where the original Alien Tiles has four states per light [17]. A comparison between the ordinary Lights Out and the two-state Alien Tiles is illustrated in Figs. 2 and 3. The two-state Alien Tiles has a neat and simple parity condition, so that this game has various discussions on the Internet such as [18,19], and also appears as training questions of programming contests [20]. The solvability and the number of solvable games of arbitrary-state Alien Tiles have been answered by Maier and Nickel [21]. However, the coding-theoretical problems that we listed above were not investigated.

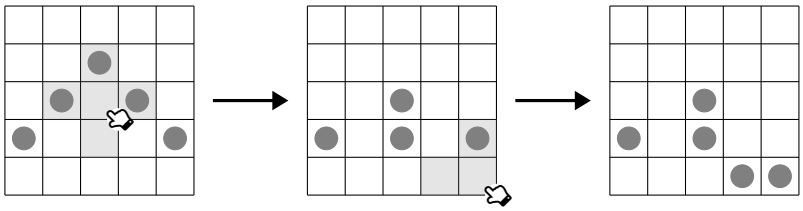


**Figure 2.** Example moves of a $5 \times 5$ Lights Out game.

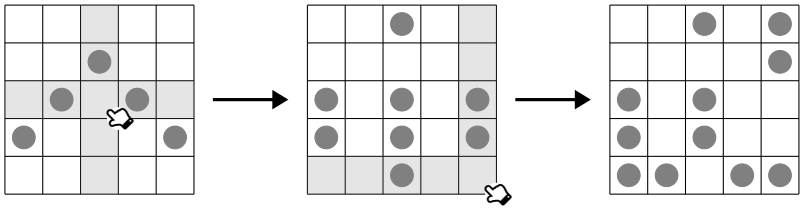


**Figure 3.** Example moves of a $5 \times 5$ two-state Alien Tiles game.

This paper is organized as follows. We first discuss some general techniques for Lights Out and its two-state variants in Section 2. Next, we discuss the basic properties of two-state Alien Tiles in Section 3. We then solve our proposed coding-theoretical problems of the two-state Alien Tiles in Section 4. In Section 5, we apply the two-state Alien Tiles as an error correction code and investigate its optimality. Lastly, we discuss a states decomposition technique to link up the original four-state Alien Tiles and the two-state variant in Section 6, and conclude this paper in Section 7. For readers who are not familiar with algorithmic complexity and coding theory, a brief introduction is provided in Sections A and B respectively.

## 2. General Techniques for Lights Out and its Two-State Variants

We now describe some existing techniques in the literature for tackling Lights Out and its two-state variants.

### 2.1. Model

For convenience, we use 1 to represent a lit light, and 0 to represent an unlit light. We consider an $m \times n$ grid of lights. Each possible grid can be represented by an $m \times n$ matrix over a binary field $\mathbb{F}_2$. A notable property of $\mathbb{F}_2$ is that addition is having the same effect as subtraction. To simplify the notations, we vectorize each $m \times n$ matrix into an $mn \times 1$ column vector by stacking the columns of the matrix on top of one another. The collection of all possible (vectorized) grids forms a vector space $G = \mathbb{F}_2^{mn}$, which is called the *game space*. When it is clear from the context, we do not vectorize the matrix form of the game for readability. In a similar manner, we can transform an arbitrary-shape game board into a vector, thus a similar vector space model also works for non-rectangular game boards.

The concept of solvability and attainability are similar. We mention both terminologies here because some literature like [21] consider attainability in lieu of solvability. A game

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



$g \in G$ is *solvable* if and only if there exists a sequence of moves to switch $g$ into a zero vector. On the other hand, a game $g \in G$ is *attainable* if and only if there exists a sequence of moves to switch a zero vector into $g$. We also denote the occurrence of the zero vector as a *solved game*.

We can represent each toggle pattern by an $m \times n$ grid of lights, where each light in the grid is on if and only if the toggle pattern toggles this light. We define the *solvable game space* by the vector space $S$ over $\mathbb{F}_2$ spanned by the set of all vectorized toggle patterns. Note that $S \subseteq G$. A game $g$ is *solvable* if and only if $g \in S$. This is due to the nature of the game that

1. applying a move twice will not toggle any lights, i.e., apply the same move again will undo the move;
2. every permutation of a sequence of moves toggles the same set of lights, i.e., the order of the moves is not taken into consideration.

It is easy to observe that solvability is equivalent to attainability in Lights Out and its two-state variants: the sequence of moves that *solves* a game can also *attain* the game.

For most Lights Out variants including the two-state Alien Tiles, we can model the moves in this manner. There are $mn$ toggle patterns in total, so there are $mn$ possible moves. We can bijectively associate each move by a light in the grid. For the two-state Alien Tiles, the $(i, j)$-th light in the grid is associated with the move that toggles all the lights in the $i$-th row and the $j$-th column. That is, this move is represented by a binary matrix where only the $(i, j)$-th entry is 1. When we perform such a move, we also say we *click* on the $(i, j)$-th light. The *move space*, denoted by $K$, is the vector space over $\mathbb{F}_2$ spanned by the set of vectorized moves, where each vector in $K$ corresponds to a sequence of (vectorized) moves, which is unique up to permutation. We remark that when there are totally $mn$ toggle patterns, $K = G$.

For a more general setting, the number of toggle patterns may not be $mn$, for example, the Gale-Berlekamp switching game. Although the physical meaning of "clicking" a light may not be valid, the concept of move space still works for this mathematical model. In such scenario, $K$ may not equal to $G$. Yet, the remaining discussions are still valid by changing certain lengths and dimensions of the vectors and the matrices.

We can relate $K$ and $G$ in the following way. Define a linear map $\psi \colon K \to G$ that outputs a game by performing the moves stated in the input on a solved game. The image of $\psi$ is $S$. Being a linear map, $\psi$ is also a homomorphism. Figure 4 illustrates two examples of mapping a $k \in K$ to a $g \in G$. We can also write $\psi$ in its matrix form $\Psi$, where $\Psi$ is an $mn \times mn$ matrix formed by juxtaposing all vectorized toggle patterns. That is, we have $\psi(k) = \Psi k$.

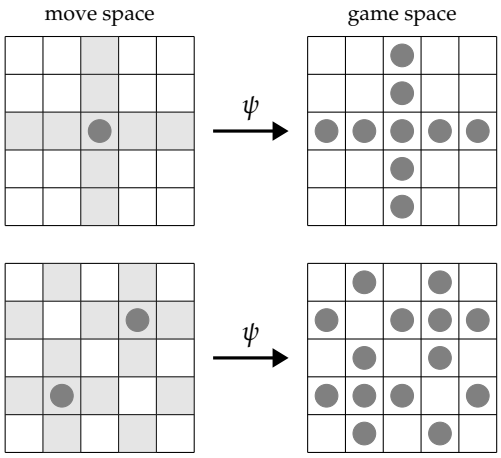



**Figure 4.** Two examples of mapping a sequence of moves in the move space $K$ to a game in the game space $G$ via the homomorphism $\psi$ for two-state Alien Tiles.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



### 2.2. Linear Algebra Approach

Suppose a $k \in K$ such that $\psi(k) = s$ for some $s \in S$ is given, then we can solve $s$ by clicking the lights lit in $k$, because $s + \psi(k)$ gives a zero vector due to the fact that subtraction is the same as addition in $\mathbb{F}_2$. In other words, solving a game $g \in G$ is equivalent to find a $k \in K$ such that $\psi(k) = g$. When we represent $\psi$ in its matrix form $\Psi$, we have a system of linear equations $\Psi k = g$. Then, we can apply the Gaussian elimination to find all possible values of $k$ such that the game is solvable.

**Definition 1** (Completely Solvable)**.** *The game is* completely solvable *if and only if $S = G$, i.e., every $g \in G$ is a solvable game.*

Completely solvable is a terminology as stated in [1]. From elementary linear algebra, we know that:

1. The game is completely solvable if and only if $\Psi$ is of full rank.
2. The orthogonal complement of $S$, denoted by $S^\perp$, is the null space of $\Psi$.

An unsolvable game, i.e., a game that is not in $S$, cannot be orthogonal to $S^\perp$. Therefore, besides brutally applying the Gaussian elimination to find $k$, we can determine the solvability of the game, by calculating $E^T g$ (where each column in $E$ corresponds to a vector in a basis of $S^\perp$). The game $s$ is said to be solvable if and only if $E^T s = 0$. This technique was also applied in [3]. In fact, this idea was also adopted in coding theory for detecting errors of a linear code.

Since $S = \{\psi(k) : k \in K\}$, where $K$ consists of all vectors in $\mathbb{F}_2^{mn}$, the dimension of the solvable game space $S$ is the same as the dimension of the vector space spanned by the columns in $\Psi$. In other words, the number of solvable games equals $2^{\text{rank}(\Psi)}$.

### 2.3. Minimal Number of Moves (The Shortest Solution)

Consider $\psi(k) = s$, i.e., the game $s$ can be solved by the move sequence $k$. Based on the property of the kernel (null space) of $\psi$, as denoted by $\ker(\psi)$, that $\psi(u) = 0$ for all $u \in \ker(\psi)$, we have $\psi(k + u) = s$ for all $u \in \ker(\psi)$ because of homomorphism.

If we want to solve $s$ with the minimal number of moves, we need to find the candidate $u$ that minimizes the 1-norm $\|k + u\|_1$. This move sequence $k + u$ is then called the *shortest solution* of the solvable game $s$. If the dimension of $\ker(\psi)$ is in terms of $m$ or $n$, then the size of $\ker(\psi)$ is exponentially large in terms of $m$ or $n$, thus an exhaustive search on all $u \in \ker(\psi)$ is inefficient.

The following theorem states the hardness of the shortest solution problem for general 2-state Lights Out variants (including the ordinary Lights Out). We adopt some terminologies in algorithmic complexity and coding theory in the proof, where their descriptions can be found in Sections A and B.

**Theorem 1.** *The shortest solution problem for general 2-state Lights Out variants is NP-hard.*

**Proof.** We prove this theorem by considering the minimum distance decoding of linear code in coding theory. As a recall, the (function) problem of minimum distance decoding of a linear code is equivalent to the following: given a parity check matrix $H$ of the linear code and a (column vector) word $k$, the goal is to find out the error (column) vector $e$ that has the smallest Hamming weight such that $H(k - e) = 0$. This is due to the property of parity check matrix that $Hc = 0$ if and only if $c$ is a codeword.

The reduction is as follows. Consider the span of the columns in $H$ as the orthogonal complement of $\ker(\psi)$, that is, every $u \in \ker(\psi)$ is a codeword. Consider $k$ as the move sequence that solves the game $s = \psi(k)$. Write $k = u + e$, with $u \in \ker(\psi)$. Since $\psi(k) = s = \psi(k - u) = \psi(e)$, we know that $e$ is a move sequence that can solve $s$. That is, by finding the shortest solution $e$ (which has the smallest Hamming weight), we have $u = k - e \in \ker(\psi)$, thus $H(k - e) = 0$. This solves the minimum distance decoding problem, and such problem is no harder than the shortest solution problem for general

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



2-state Lights Out variants. As minimum distance decoding is NP-hard [7,8], the shortest solution problem for general 2-state Lights Out variants is also NP-hard. □

Nevertheless, it is still possible to have an efficient algorithm for finding the shortest solution of a specific Lights Out variant. For example, if the game space is completely solvable, then we must have $\|e\|_1 = 0$. In such case, the move sequence obtained by the Gaussian elimination process is unique, so this problem is in P (in cubic time) complexity.

As a remark, we can also reduce this shortest solution problem to the minimum distance decoding problem in a similar manner, by treating the juxtaposition of the (column) vectors in the basis of the orthogonal complement of $\ker(\psi)$ as the parity check matrix $H$. This shows that the two problems are actually equivalent.

## 3. Two-State Alien Tiles

We now discuss some specific techniques for two-state Alien Tiles. Note that we can regard the game space $G$ and the solvable game space $S$ as abelian groups under vector addition. Then, we have $S \lhd G$, i.e., $S$ is a normal subgroup of $G$.

**Definition 2** (Syndrome)**.** *Two games have the same* syndrome *if and only if they are in the same coset in the quotient group $G/S$.*

One crucial component that will be applied into solving our coding-theoretical problems is the function that outputs the syndrome of a game. The quotient group $G/S$ is isomorphic with the set of all syndromes.

We adopt the term "syndrome" in coding theory here because each coset represents an "error" from the solvable game space that cannot be recovered by any move sequences. In other words, an unsolvable game becomes solvable after we remove the "error" by toggling some individual lights.

### *3.1. Easy Games and Doubly Easy Games*

**Definition 3** (Easy Games)**.** *A game $s$ is called* easy *if and only if it can be solved by clicking the lights lit in $s$, i.e., $\psi(s) = s$.*

The terminology "easy" was due to Torrence [22]. In other words, $s$ is easy if $\mathrm{vec}(s)$ is an eigenvector of $\Psi$. We also extend the terminology as follows.

**Definition 4** (Doubly Easy Games)**.** *A game $s$ is* doubly easy *if and only if it can be solved by first clicking the lights lit in $s$ to obtain a game $s + \psi(s)$, then clicking the lights lit in the game $s + \psi(s)$.*

That is, a game $s$ is doubly easy if and only if

$$s + \psi(s) + \psi(s + \psi(s)) = 0,$$

which is equivalent to $\psi(\psi(s)) = s$ by the definition of homomorphism.

To write down $\Psi$ explicitly, we consider the following. Denote by $\mathbb{1}_{a,b}$ and $\mathbb{1}_c$ an $a \times b$ and a $c \times c$ all-ones matrix respectively. Similarly, denote by $\mathbf{0}_{a,b}$ and $\mathbf{0}_c$ an $a \times b$ and a $c \times c$ zero matrix respectively. Let $\mathbf{I}_c$ be a $c \times c$ identity matrix. By juxtaposing all (vectorized) toggle patterns, we obtain an $mn \times mn$ matrix

$$\Psi = \begin{pmatrix} \mathbb{1}_m & \mathbf{I}_m & \mathbf{I}_m & \cdots & \mathbf{I}_m \\ \mathbf{I}_m & \mathbb{1}_m & \mathbf{I}_m & \cdots & \mathbf{I}_m \\ \mathbf{I}_m & \mathbf{I}_m & \mathbb{1}_m & \cdots & \mathbf{I}_m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{I}_m & \mathbf{I}_m & \mathbf{I}_m & \cdots & \mathbb{1}_m \end{pmatrix} = \mathbb{1}_n \otimes \mathbf{I}_m + \mathbf{I}_n \otimes \mathbb{1}_m - \mathbf{I}_{mn}.$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



To analyze doubly easy games, we need to apply $\psi$ multiple times on a given $k \in K$. For any positive integer $c$ and matrix $A$ over $\mathbb{F}_2$, we write $cA$ to denote $(c \bmod 2)A$ for simplicity. Also, note that

$$\mathbb{1}_c^2 = \begin{cases} \mathbf{0}_c & \text{if } c \text{ is even,} \\ \mathbb{1}_c & \text{if } c \text{ is odd.} \end{cases}$$

In other words, $\mathbb{1}_c^2 = c\mathbb{1}_c$. By making use of the mixed-product property of Kronecker product:

$$(A \otimes B)(C \otimes D) = (AC) \otimes (BD)$$

where $A$, $B$, $C$ and $D$ are matrices of suitable sizes that can form matrix products $AC$ and $BD$, we can now evaluate the powers of $\Psi$ (over $\mathbb{F}_2$).

$$\begin{aligned}
\Psi^2 &= \mathbb{1}_n^2 \otimes \mathbf{I}_m + \mathbb{1}_n \otimes \mathbb{1}_m - \mathbb{1}_n \otimes \mathbf{I}_m + \mathbb{1}_n \otimes \mathbb{1}_m + \mathbf{I}_n \otimes \mathbb{1}_m^2 - \mathbf{I}_n \otimes \mathbb{1}_m \\
&\quad - \mathbb{1}_n \otimes \mathbf{I}_m - \mathbf{I}_n \otimes \mathbb{1}_m + \mathbf{I}_{mn} \\
&= n\mathbb{1}_n \otimes \mathbf{I}_m + m\mathbf{I}_n \otimes \mathbb{1}_m - \mathbf{I}_{nm}, \\
\Psi^3 &= n(\mathbb{1}_n^2 \otimes \mathbf{I}_m + \mathbb{1}_n \otimes \mathbb{1}_m - \mathbb{1}_n \otimes \mathbf{I}_m) + m(\mathbb{1}_n \otimes \mathbb{1}_m + \mathbf{I}_n \otimes \mathbb{1}_m^2 - \mathbf{I}_n \otimes \mathbb{1}_m) \\
&\quad - \mathbb{1}_n \otimes \mathbf{I}_m - \mathbf{I}_n \otimes \mathbb{1}_m + \mathbf{I}_{mn} \\
&= n(n-1)\mathbb{1}_n \otimes \mathbf{I}_m + m(m-1)\mathbf{I}_n \otimes \mathbb{1}_m - \mathbb{1}_n \otimes \mathbf{I}_m - \mathbf{I}_n \otimes \mathbb{1}_m + \mathbf{I}_{mn} + (m+n)\mathbb{1}_{mn} \\
&= \mathbb{1}_n \otimes \mathbf{I}_m + \mathbf{I}_n \otimes \mathbb{1}_m - \mathbf{I}_{mn} + (m+n)\mathbb{1}_{mn} \\
&= \Psi + (m+n)\mathbb{1}_{mn}, \\
\Psi^4 &= \Psi^2 + (m+n)\Psi\mathbb{1}_{mn} \\
&= \Psi^2 + (m+n)(m+n-1)\mathbb{1}_{mn} \\
&= \Psi^2.
\end{aligned}$$

That is, for any positive integer $c \geq 4$, we have $\Psi^c = \Psi^{2+(c \bmod 2)}$.

### 3.2. Even-by-Even Games

When both $m$ and $n$ are even, we have $\Psi^2 = \mathbf{I}_{mn}$. In other words, $\Psi^{-1} = \Psi$ and $\psi$ is an isomorphism, thus $S = K = G$. This means that all games are solvable, i.e., every even-by-even 2-state Alien Tiles is completely solvable. Therefore, the number of solvable games is $2^{mn}$.

Consider an arbitrary solvable game $s \in S$. There exists a unique $k \in K$ such that $\psi(k) = s$. Then, we have $k = \psi^{-1}(s) = \psi(s)$, thus $\psi(\psi(s)) = s$. In other words, all solvable games are doubly easy. The size of $\ker(\psi)$ is 1 as it is a zero vector space. This result for square grids was also discussed in [23]. An interesting property of even-by-even games is the duality that $k \in K$ solves the game $s \in S$ if and only if $s \in K$ solves the game $k \in S$.

We can also view the solvability in another manner. If we know how to toggle an arbitrary light, we can repeatedly apply this strategy to switch off the lights one by one. This also means that all games are solvable. We can toggle a single light by all the lights in the same row and the same column (where the light at the intersection of the row and the column is clicked once only). This can be observed from $\psi(\psi(s)) = s$ when $s$ only consists of a single 1.

To compute $k = \psi^{-1}(s) = \psi(s)$, the worst case (i.e., the case that takes the maximal number of computational steps) occurs when $s$ is an all-ones game (a game that has all lights lit, which is an all-ones column vector of length $mn$). Every 1 in $s$ toggles $m + n - 1$ lights. Therefore, the complexity to compute $k$ is $\mathcal{O}(mn(m+n-1)) = \mathcal{O}(m^2n + mn^2)$, which has the same complexity as solving the linear system $\Psi k = s$ directly by Gaussian elimination. On the other hand, due to the uniqueness of $k$, the minimal number of moves

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



to solve $s$ is $\|k\|_1 = \|\psi(s)\|_1$. To compute $\|k\|_1$, we scan the status of the lights after we obtain $k$, thus the complexity is $\mathcal{O}(mn(m+n-1)+mn) = \mathcal{O}(m^2n+mn^2)$.

### 3.3. Even-by-Odd (and Odd-by-Even) Games

Due to symmetry, we only need to discuss even-by-odd games. We first discuss how to solve these games.

Note that $\Psi^3 = \Psi + \mathbb{1}_{mn}$. For each solvable game $s \in S$, pick an arbitrary $k \in K$ such that $\psi(k) = s$. Since $\psi(\psi(\psi(k))) = \psi(\psi(s))$, by applying the $\psi$ map twice, we have

$$\psi(\psi(s)) = \psi(k) + \mathbb{1}_{mn}k = s + \|k\|_1 \mathbb{1}_{mn,1}.$$

By clicking the lit lights in $s$, we obtain the game $s + \psi(s)$. Afterwards, we click the lit lights in $s + \psi(s)$ to obtain the game

$$s + \psi(s) + \psi(s + \psi(s)) = s + \psi(\psi(s)) = \|k\|_1 \mathbb{1}_{mn,1},$$

which is either a solved game or an all-ones game.

To see how the all-ones game is being solved, we consider the following: An arbitrary row of odd length $n$ is selected, and every light in this row is clicked. In this way, every light in this row is toggled $n$ times, and every light not belonging to this row is toggled once. Note that $n \bmod 2 = 1$, so every light in the game is toggled once. In other words, this move sequence can solve the all-ones game.

The worst case complexity to compute $s + \psi(s)$ is $\mathcal{O}(m^2n + mn^2)$. Given $s + \psi(s)$, we have the same complexity for computing $s + \psi(s) + \psi(s + \psi(s))$. The complexity to click every light in an arbitrary row is $\mathcal{O}(mn^2)$. Therefore, the overall complexity to solve a solvable game is $\mathcal{O}(m^2n + mn^2)$. We have the same complexity for finding the move sequence because the complexity for checking whether $\|k\|_1 \bmod 2 = 1$ is $\mathcal{O}(mn)$, which is absorbed into $\mathcal{O}(m^2n + mn^2)$.

Next, we discuss the move sequence to toggle all the lights in an arbitrary column, which will be used in the remaining discussion. Suppose we want to toggle the $j$-th column. Consider this move sequence: Except the $(1, j)$-th light, click every light in the $j$-th column and every light in the 1-st row. Based on such operations, we have the followings:

- The $(1, j)$-th light is toggled for $m + n - 2$ times, where $m + n - 2$ is odd.
- Each of the other lights in the $j$-th column is toggled for $m - 1$ times, where $m - 1$ is odd.
- Each of the other lights in the 1-st row is toggled for $n - 1$ times, where $n - 1$ is even.
- All other lights that are not in the $j$-th column and not in the $i$-th row are toggled twice.

As a result, only the lights in the $j$-th column is toggled for odd number of times, then all the lights in the $j$-th column are being toggled.

We notice that if we apply the above move sequence to every column, the resultant move sequence is that except the first row, all the other lights are clicked once. Although this can solve the all-ones game, this move sequence is longer than what we have previously described. This suggests that the move sequence for solving a solvable game is not unique.

We now discuss the method to check whether a game is solvable or not. Denote by $g_{ij}$ the $(i, j)$-th light in the game $g$. Let $r_i(g)$ be the row parity of the $i$-th row in $g$, i.e.,

$$r_i(g) = \left( \sum_{j=1}^{n} g_{ij} \right) \bmod 2.$$

Let $r(g) = (r_1(g), r_2(g), \ldots, r_m(g))^\intercal$ be a column vector over $\mathbb{F}_2$, called the *row parity vector*. We can consider this row parity vector as exclusive or-ing (XOR) in all the columns, or the nim-sum of the columns.

Suppose we click an arbitrary light, say, the $(y, x)$-th light. Then, the values of $g_{ij}$ at all $i = y$ or $j = x$ are toggled. We can see that the new $r_y$ satisfies $r_y + n = r_y + 1$. Except

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



when $i = y$, the new $r_i$ will become $r_i + 1$. That is, all parity bits in $r$ are flipped. This is illustrated in Fig. 5.

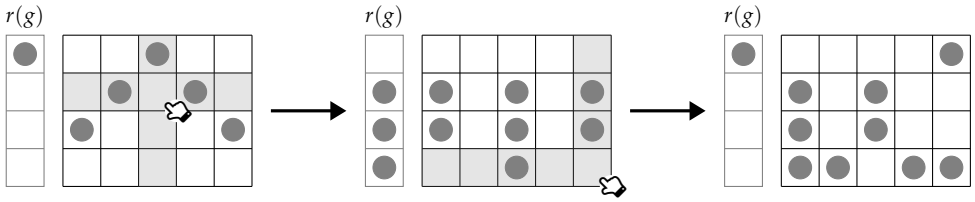



**Figure 5.** The row parity vector is flipped after performing a move.

To map both row parity vectors into one invariant such that any move sequence acting on a game does not change the invariant, we define the invariant function $\tilde{r}\colon G \to \mathbb{F}_2^{(m-1)\times 1}$ by

$$\begin{aligned}\tilde{r}(g) &= (r_2(g), r_3(g), \ldots, r_m(g))^\intercal + r_1(g)\mathbb{1}_{m-1,1} \\ &= \begin{cases} (r_2(g), r_3(g), \ldots, r_m(g))^\intercal & \text{if } r_1(g) = 0, \\ (r_2(g), r_3(g), \ldots, r_m(g))^\intercal + \mathbb{1}_{m-1,1} & \text{if } r_1(g) = 1. \end{cases}\end{aligned}$$

As every solvable game $s \in S$ gives $\tilde{r}(s) = \mathbf{0}_{m-1,1}$, we have $\|s\|_1 \bmod 2 = \|r(s)\|_1 \bmod 2 = 0$, i.e., the number of lit lights in a solvable game is even. Also, $\tilde{r}(g) \neq \mathbf{0}_{m-1,1}$ implies that the game $g$ is not solvable. This suggests that not all games are solvable, say, a game where only the $(1,1)$-st light is lit has an invariant $\mathbb{1}_{m-1,1} \neq \mathbf{0}_{m-1,1}$.

On the other hand, any move sequence acting on a game in a coset $C \in G/S$ gives a game in the same coset, because $S$ is the span of all toggle patterns. Then, the following question arises: What is the relation between the invariant function and the cosets? To be more specific: Does the invariant function give a syndrome of the game? The answer is affirmative.

**Theorem 2.** *Let $g$ be an even-by-odd game. $\tilde{r}$ is a surjective map that maps $g$ to its syndrome.*

**Proof.** See Section C. □

The above theorem has the following consequences:

1. $\tilde{r}(g) = \mathbf{0}_{m-1,1}$ if and only if $g \in S$, because $\mathbf{0}_{m-1,1}$ is the syndrome of the solved game.
2. The number of cosets in $G/S$, i.e., the index $[G : S]$, is $2^{m-1}$, due to the surjection of $\tilde{r}$.
3. The number of solvable games, i.e., $|S|$, is $|G|/[G : S] = 2^{mn-m+1}$ according to Lagrange's theorem (in group theory).
4. A game is solvable if and only if the parities of all rows are the same. The complexity of verifying the solvability is thus $\mathcal{O}(mn)$.

The method to find the minimal number of moves for solving an even-by-odd game efficiently is remained as an open problem. To justify that exhaustive search for this problem is inefficient, we calculate the size of $\ker(\psi)$. By the rank-nullity theorem, we know that $\dim(\ker(\psi)) = m - 1$. Therefore, $|\ker(\psi)| = 2^{m-1}$, which means that exhaustive search cannot be done in polynomial time.

### *3.4. Odd-by-Odd Games*

When both $m$ and $n$ are odd, we have $\Psi^2 = \Psi$. In other words, we have $\psi \circ \psi = \psi$. For each solvable game $s \in S$, pick an arbitrary $k \in K$ such that $\psi(k) = s$. By applying $\psi$ again, we obtain $\psi(\psi(k)) = \psi(s)$. Using the fact that $\psi(\psi(k)) = \psi(k)$, we have $\psi(s) = \psi(k) = s$. That is, all solvable games are easy. Similar as the above subsections, the worst case

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



complexity to solve the game is $\mathcal{O}(m^2n + nm^2)$. To find a move sequence $k$, we only need to output $s$ directly, thus the complexity is $\mathcal{O}(mn)$.

To check whether a game $g$ is solvable or not, it is insufficient to only consider the row parity vector $r(g)$. Let $c_j(g)$ be the column parity of the $j$-th column in $g$, i.e.,

$$c_j(g) = \left(\sum_{i=1}^{m} g_{ij}\right) \bmod 2,$$

and let $c(g) = (c_1(g), c_2(g), \ldots, c_n(g))$ be a row vector over $\mathbb{F}_2$, called the *column parity vector*, when we click an arbitrary light, say, the $(y, x)$-th light, we can see that the new $r_y$ becomes $r_y + n = r_y + 1$, and the new $c_x$ is $c_x + m = c_x + 1$. Except when $(i, j) = (y, x)$, where the new $r_i$ is $r_i + 1$ and the new $c_j$ is $c_j + 1$. In that case, all parity bits of $r(g)$ and $c(g)$ are flipped. This is illustrated in Fig. 6.

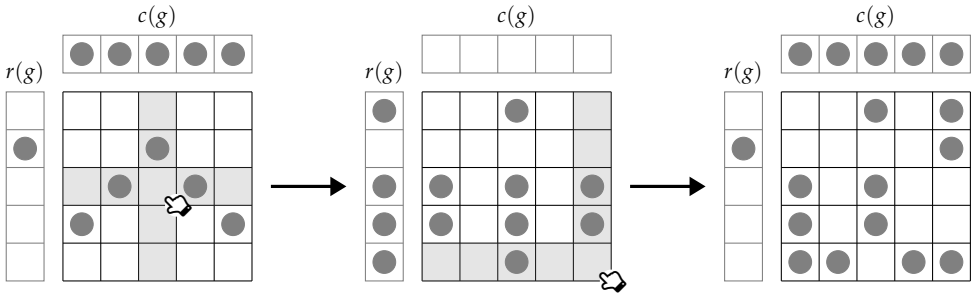



**Figure 6.** Both the row and column parity vectors are flipped after performing a move.

We call the pair $(r(g), c(g))$ the *parity pair* of the game $g$. We remark that not all combinations of parity pairs are valid. The exact criteria is stated in the theorem below.

**Theorem 3.** *For odd-by-odd games, $g \in G$ if and only if $\|r(g)\|_1 - \|c(g)\|_1$ is even.*

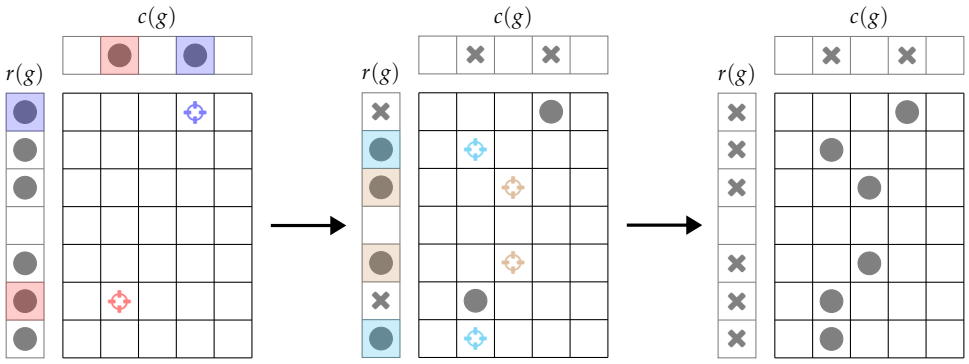
**Proof.** We first prove the "only if" part. The result is trivial for the solved game $g = \mathbf{0}_{mn,1}$. Consider an arbitrary non-zero game $g \in G \setminus \{\mathbf{0}_{mn,1}\}$. When we toggle an arbitrary lit light in $g$, say, the $(y, x)$-th light to form a new game $g'$, the row parity of the $y$-th row and the column parity of the $x$-th column are both flipped. That is, $(\|r(g')\|_1 - \|c(g')\|_1) \bmod 2$ equals $(\|r(g)\|_1 - \|c(g)\|_1) \bmod 2$. Inductively, the procedure is repeated until all lit lights are off. Then, $(\|r(\mathbf{0}_{mn,1})\|_1 - \|c(\mathbf{0}_{mn,1})\|_1) \bmod 2$ is equivalent to $(\|r(g)\|_1 - \|c(g)\|_1) \bmod 2$, where the former one equals 0. In other words, $g \in G$ implies that $\|r(g)\|_1 - \|c(g)\|_1$ is even.

For the "if" part, we prove by construction. Let $\{y_1, y_2, \ldots, y_{\|r(g)\|_1}\}$ be an index set such that $r_y(g) = 1$ if and only if $y$ is in this set. Similarly, let $\{x_1, x_2, \ldots, x_{\|c(g)\|_1}\}$ be an index set such that $c_x(g) = 1$ if and only if $x$ is in this set.

First, we construct a game such that only the $(y_i, x_i)$-th lights are lit, where $i = 1, 2, \ldots, \min\{\|r(g)\|_1, \|c(g)\|_1\}$. If $\|r(g)\|_1 = \|c(g)\|_1$, then the proof is done. If $\|r(g)\|_1 > \|c(g)\|_1$, let $T = \{y_{\|c(g)\|_1+1}, y_{\|c(g)\|_1+2}, \ldots, y_{\|r(g)\|_1}\}$. Then, we partition $T$ into sets of size 2, denoted by $\left\{P_1, P_2, \ldots, P_{\frac{\|r(g)\|_1 - \|c(g)\|_1}{2}}\right\}$. For every $i = 1, 2, \ldots, \frac{\|r(g)\|_1 - \|c(g)\|_1}{2}$, let $P_i = \{a_i, b_i\}$, and we switch on the $(a_i, j)$-th and the $(b_i, j)$-th lights for an arbitrary $j \in \{1, 2, \ldots, n\}$. Afterwards, the row parity vector of the game matches with $r(g)$. For each partition, we have switched on 2 lights in the same column, so the column parity of this column remains unchanged. That is, the parity pair of the resultant game is $(r(g), c(g))$. We can prove the case $\|c(g)\|_1 > \|r(g)\|_1$ in a similar manner by symmetry arguments. The existence of this game in the prescribed game space. □

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1






**(a)** Bit-matching phase. **(b)** Bit-pairing phase. **(c)** Constructed game.

**Figure 7.** An example of constructing a game from the parity pairs.

The construction in the above proof will also be applicable when we discuss the coding-theoretical problems. We illustrate an example of such construction in Fig. 7. In this example, we start from a solved game, i.e., all lights are off. In the first phase, we match the bits in the row parity vector to the column parity vector. This example has fewer bits in the column parity vector, thus all the bits in the column parity vector are matched with some bits in the row parity vector. The bits in the row parity vector that are matched are arbitrary. In Section 3.4, the red bits are matched, followed by the blue bits. Each pair of matched bits indicates a location, which is marked with a crosshair of the same color in the game grid. By switching on the lights marked by the crosshairs, the second phase is proceeded.

The bits that are not matched must all fall in either the row or the column parity vector. In this example, there are four bits left in the row parity vector. We group the bits two by two arbitrarily. In Section 3.4, the two cyan bits and the two brown bits are respectively grouped. For each group, we can select an arbitrary column (if the unmatched bits fall in the column parity vector), followed by the selection of an arbitrary row. The crosshairs of the same color mark the column selected by the same group. By switching on the lights marked by the crosshairs, the desired game is constructed.

We now discuss the invariant function. Similar to the $\tilde{r}$ in even-by-odd games, we define $\tilde{c}\colon G \to \mathbb{F}_2^{1\times(n-1)}$ by

$$\begin{aligned}\tilde{c}(g) &= (c_2(g), c_3(g), \ldots, c_n(g)) + c_1(g)\mathbb{1}_{1,n-1} \\ &= \begin{cases} (c_2(g), c_3(g), \ldots, c_n(g)) & \text{if } c_1(g) = 0, \\ (c_2(g), c_3(g), \ldots, c_n(g)) + \mathbb{1}_{1,n-1} & \text{if } c_1(g) = 1. \end{cases}\end{aligned}$$

The invariant function for odd-by-odd games, denoted by $\mathrm{inv}\colon G \to \mathbb{F}_2^{(m-1)\times 1} \times \mathbb{F}_2^{1\times(n-1)}$, is defined as

$$\mathrm{inv}(g) = (\tilde{r}(g), \tilde{c}(g)).$$

Again, any move sequence acting on a game will never alter the invariant. Therefore, $\mathrm{inv}(g) \neq (\mathbf{0}_{m-1,1}, \mathbf{0}_{1,n-1})$ implicates that $g$ is not solvable. In particular, a game with only the $(1,1)$-st light lit has an invariant $(\mathbb{1}_{m-1,1}, \mathbb{1}_{1,n-1}) \neq (\mathbf{0}_{m-1,1}, \mathbf{0}_{1,n-1})$.

Unlike the case of even-by-odd games, the number of lit lights in a solvable game $s \in S$ can be either odd or even. For example, an all-ones game has odd number of lit lights, which can be solved by clicking all the lights so that every light is toggled for $m+n-1$ times, where $m+n-1$ is odd. On the other hand, a solved game is a solvable game with no lit lights, hence the number of lit lights is even.

**Theorem 4.** *Let $g$ be an odd-by-odd game.* inv *is a surjective function that maps $g$ to its syndrome.*

**Proof.** See Section D. □

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



Similar to the discussion in the last subsection, the consequences of the above theorem include:

1. $\mathrm{inv}(g) = (\mathbf{0}_{m-1,1}, \mathbf{0}_{1,n-1})$ if and only if $g \in S$, because $(\mathbf{0}_{m-1,1}, \mathbf{0}_{1,n-1})$ is the syndrome of the solved game.
2. The number of cosets in $G/S$, i.e., the index $[G : S]$ is $2^{m+n-2}$, due to the surjection of inv.
3. The number of solvable games, i.e., $|S|$, is $|G|/[G : S] = 2^{mn-m-n+2}$ according to Lagrange's theorem.
4. A game is solvable if and only if the parities of all rows and all columns are the same. The complexity of verifying its solvability is thus $\mathcal{O}(mn)$.

Seeking for an efficient algorithm to find the minimal number of moves in solving an odd-by-odd game remains an open problem nowadays. For square grids, i.e., when $n = m$, the simplified problem was discussed in [24,25], but the answers there for odd-by-odd grids are certain exhaustive searches, which have to undergo the trial of $2^{2n-2}$ possibilities, where this number equals to the size of $\ker(\psi)$ for odd-by-odd square grids as discussed in [23]. For the general sense, we can easily show that $|\ker(\psi)| = 2^{m+n-2}$ by the rank-nullity theorem. This means that an exhaustive search on the kernel cannot be efficiently conducted.

## 4. Coding-Theoretical Perspective of Two-State Alien Tiles

The solvable game space $S$ is a vector subspace of the game space $G$. That is, we can regard $S$ as a linear code. Our task here is to find out the properties of $S$ from the perspective of coding theory. We refer our readers to Section B for a brief introduction to coding theory.

### *4.1. Hamming Weight*

The basic parameter of an error correction code is the *distance* of the code. When the distance $d$ is known, the error detecting ability $d-1$ and the error correcting ability $\lfloor \frac{d-1}{2} \rfloor$ are then obtained. As a linear code, the distance of the code equals the *Hamming weight* of the code, which is defined as the minimum Hamming weight among all the non-zero codewords of the code. However, finding the Hamming weight of a general linear code is NP-hard [16]. For two-state Alien Tiles, we can determine the Hamming weight of the solvable game space with ease, via the use of invariant functions.

**Even-by-Even Games**. These games are completely solvable, so a game with only one lit light has the minimum Hamming weight among all non-zero solvable game, i.e., the Hamming weight of such code is 1. This also means that there is no error detecting and correction abilities for even-by-even games.

**Even-by-Odd Games**. The syndrome of an even-by-odd solvable game is $\mathbf{0}_{m-1,1}$. There are two possible row parity vectors (of size $m \times 1$) that correspond to this syndrome, namely $\mathbf{0}_{m,1}$ and $\mathbb{1}_{m,1}$.

We first consider the row parity vector $\mathbf{0}_{m,1}$. Each row has an even number of lit lights. As the Hamming weight of a code only considers non-zero codewords, we cannot have a row parity vector $\mathbf{0}_{m,1}$ for any non-zero solvable game if $n = 1$. For $n > 1$, the smallest $\|g\|_1$ equals to 2, where $\mathbf{0}_{mn,1} \neq g \in S$ and $r(g) = \mathbf{0}_{m,1}$.

Now, we consider the row parity vector $\mathbb{1}_{m,1}$. Each row has an odd number of lit lights. That is, the smallest $\|g\|_1$ equals to $m$, where $\mathbf{0}_{mn,1} \neq g \in S$ and $r(g) = \mathbb{1}_{m,1}$. As $m > 0$ is even, the smallest possible $m$ is 2. That is, we do not need to consider the row parity vector $\mathbb{1}_{m,1}$ unless $n = 1$.

As a summary, we have:

$$\text{For even-by-odd games,} \quad \min_{g \in S \setminus \{\mathbf{0}_{mn,1}\}} \|g\|_1 = \begin{cases} m & \text{if } n = 1, \\ 2 & \text{otherwise.} \end{cases}$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



**Odd-by-Odd Games**. The syndrome of an odd-by-odd solvable game is the pair $(\mathbf{0}_{m-1,1}, \mathbf{0}_{1,n-1})$. If $m = 1$, only two games are in $S$, which are the all-zeros and the all-ones games. Therefore, the Hamming weight of the code is $n$. If $n = 1$, similarly, the Hamming weight is $m$. Both cases hold at the same time when $m = n = 1$.

Now, consider the case with both $m$ and $n$ greater than 1, i.e., each of them is at least 3. There are two possible parity pairs (in $\mathbb{F}_2^{m\times 1} \times \mathbb{F}_2^{1\times n}$), namely $(\mathbf{0}_{m,1}, \mathbf{0}_{1,n})$ and $(\mathbb{1}_{m,1}, \mathbb{1}_{1,n})$. For the former case, each row and each column has an even number of lit lights. In other words, the smallest $\|g\|_1$, where $\mathbf{0}_{mn,1} \neq g \in S$ and $(r(g), c(g)) = (\mathbf{0}_{m,1}, \mathbf{0}_{1,n})$, is 4. For the latter case, each row and each column has an odd number of lit lights. That is, the minimal number of lit lights is no fewer than $\max\{m, n\}$.

When $m = n$, it is easy to see that the minimal number of lit lights can be achieved by a game represented by an identity matrix. If $m \neq n$, without loss of generality, consider the case of $n > m$. An example to achieve the minimal number of lit lights $\max\{m, n\}$ is the game

$$\left(\mathbf{I}_m \;\middle|\; \begin{matrix}\mathbf{0}_{m-1,n-m} \\ \hline \mathbb{1}_{1,n-m}\end{matrix}\right).$$

Note that unless $m = n = 3$, we have $\max\{m, n\} \geq 5$. Therefore, we reach the conclusion that:

$$\text{For odd-by-odd games,} \qquad \min_{g\in S\setminus\{\mathbf{0}_{mn,1}\}} \|g\|_1 = \begin{cases} n & \text{if } m = 1, \\ m & \text{if } n = 1, \\ 3 & \text{if } m = n = 3, \\ 4 & \text{otherwise.} \end{cases}$$

One may further combine the aforementioned results of all game sizes as follows:

$$\min_{g\in S\setminus\{\mathbf{0}_{mn,1}\}} \|g\|_1 = \begin{cases} n & \text{if } m = 1, \\ m & \text{if } n = 1, \\ 3 & \text{if } m = n = 3, \\ (1 + (m \bmod 2))(1 + (n \bmod 2)) & \text{otherwise.} \end{cases}$$

### *4.2. Coset Leader*

The quotient group $G/S$ contains all the cosets, where each coset contains all games achieved by any possible finite move sequence from a certain setup of the game. Two games are in the same coset if and only if there exists a move sequence to transform one to another. We have discussed the technique to identify the coset that a game belongs to by finding the syndrome of the game, where each coset is associated with a unique syndrome.

Any game $g \in C \in G/S$, with $C \neq S$ being not solvable. In the view of coding theory, we are interested in the minimal number of independent lights that we need to toggle such that the game becomes solvable? In algebraic point of view, we want to know how far the coset $C$ and the solvable space $S$ are separated. Mathematically speaking, the mission is to find a game $e \in C$ with smallest $\|e\|_1$ such that $g + e \in S$. Such $e$ can be regarded as an error, so if we know the error pattern of each coset, an error correction code can be applied. In coding-theoretical terminology, this $e$ is called the *coset leader* of the coset $C$.

**Even-by-Even Games**. As even-by-even games have no error correcting ability, it is not an interesting problem to find the coset leader. In fact, we have $G/S = \{S\} = \{G\}$, thus the coset leader of the only coset is the solved game $\mathbf{0}_{mn,1}$, which contains no lit lights.

**Even-by-Odd Games**. We consider the syndrome $\tilde{r}(g) = (\tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\intercal \in \mathbb{F}_2^{(m-1)\times 1}$ of a game $g$ in a coset $C$. We have two possible row parity vectors, namely

$$r = (0, \tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\intercal \quad \text{and} \quad r' = r + \mathbb{1}_{m,1}.$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



First, note that the row parities of different rows are independent of each other. If the row parity of a row is 0, then there are even number of lit lights in this row. That is, the minimal number of lit lights in this row is 0. If the row parity is 1, then there are odd number of lit lights in this row, where the minimal number is 1. In other words, the smallest $\|g\|_1$ such that $r(g) = r$ is $\|r\|_1$. The situation is similar for $r'$.

Note that $\|\tilde{r}(g)\| = \|r\|_1$ and $\|r\|_1 = m - \|r'\|_1$. As both $r$ and $r'$ correspond to the same syndrome $\tilde{r}(g)$, we know that

$$\min_{g \in C} \|g\|_1 = \min\{\|r\|_1, \|r'\|_1\} = \min\{\|\tilde{r}(g)\|_1, m - \|\tilde{r}(g)\|_1\},$$

which is of time complexity $\mathcal{O}(m)$ when $\tilde{r}(g)$ is given. The coset leader is an arbitrary game that belongs to $\arg\min_{g \in C} \|g\|_1$, for example,

$$\begin{cases} \left( \begin{array}{c|c} 0 & \\ \tilde{r}(g) & \mathbf{0}_{m,n-1} \end{array} \right) & \text{if } \|\tilde{r}(g)\|_1 \leq m - \|\tilde{r}(g)\|, \\ \left( \begin{array}{c|c} 1 & \\ \tilde{r}(g) + \mathbb{1}_{m-1,1} & \mathbf{0}_{m,n-1} \end{array} \right) & \text{otherwise.} \end{cases}$$

The construction can be completed within $\mathcal{O}(mn)$ time. During the construction of the coset leader, if we need to put a bit in a row, we can locate it arbitrarily. Therefore, the number of coset leaders in the coset is $n^{\min\{\|\tilde{r}(g)\|_1, m - \|\tilde{r}(g)\|_1\}}$.

**Odd-by-Odd Games**. Consider the syndrome $\text{inv}(g) = (\tilde{r}(g), \tilde{c}(g))$. Write $\tilde{r}(g) = (\tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\intercal$ and $\tilde{c}(g) = (\tilde{c}_2, \tilde{c}_3, \ldots, \tilde{c}_n)$. Let

$$\bar{r} = (0, \tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\intercal \quad \text{and} \quad \bar{c} = (0, \tilde{c}_2, \tilde{c}_3, \ldots, \tilde{c}_n).$$

The syndrome can be induced by four parity pairs, namely

$$(\bar{r}, \bar{c}), (\bar{r} + \mathbb{1}_{m,1}, \bar{c}), (\bar{r}, \bar{c} + \mathbb{1}_{1,n}) \text{ and } (\bar{r} + \mathbb{1}_{m,1}, \bar{c} + \mathbb{1}_{1,n}).$$

According to Theorem 3, we know that only two out of the four parity pairs are valid, because the difference between the number of bits in the row and the column parity vectors must be an even number. Let $(r, c)$ be a valid parity pair. If the parity of a row/column is 0, then the minimal number of lit lights in this row/column is 0. If the parity is 1, then the minimal number is 1. Thus, the minimal number of lit lights is bounded below by $\max\{\|r\|_1, \|c\|_1\}$.

Now, we use the construction in the proof of Theorem 3 to construct a game $g'$ with parity pair $(r, c)$. Note that the number of lit lights in $g'$ is $\max\{\|r\|_1, \|c\|_1\}$, which matches with the lower bound. As there are two valid parity pairs, we need to see which one results in a game that has fewer lit lights. The one with the fewer lit lights is the coset leader.

The overall procedure to find the coset leader is as follows:

1. Generate $(r, c) = (\bar{r}, \bar{c})$ from $\text{inv}(g)$.
2. If $\|r\|_1 - \|c\|_1$ is not an even number, flip all the parity bits in either $r$ or $c$.
3. If $\max\{m - \|r\|_1, n - \|c\|_1\} < \max\{\|r\|_1, \|c\|_1\}$, then flip all the parity bits in both $r$ and $c$.
4. Construct a game that has a parity pair $(r, c)$ (by using the construction in the proof of Theorem 3), and such game is denoted as the coset leader.

As we need $\mathcal{O}(mn)$ time to construct such game, the complexities of other computation steps that take either $\mathcal{O}(m)$ or $\mathcal{O}(n)$ are being absorbed by this dominant term, thus the overall complexity to construct a coset leader is $\mathcal{O}(mn)$.

At the end of this construction, we can also obtain the minimal number of lit lights among all games in the coset, which is $\max\{\|r\|_1, \|c\|_1\}$. Let

$$P = (\|\tilde{r}(g)\|_1 - \|\tilde{c}(g)\|_1) \bmod 2 \in \{0, 1\}.$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



We can also directly calculate this number as follows, with $c$ being flipped:

$$\min\Big\{\max\{\|\tilde{r}(g)\|_1, Pn+(-1)^P\|\tilde{c}(g)\|_1\}, \max\{m-\|\tilde{r}(g)\|_1, (1-P)n+(-1)^{1-P}\|\tilde{c}(g)\|_1\}\Big\},$$

or equivalently (flipping $r$),

$$\min\Big\{\max\{Pm+(-1)^P\|\tilde{r}(g)\|_1, \|\tilde{c}(g)\|_1\}, \max\{(1-P)m+(-1)^{1-P}\|\tilde{r}(g)\|_1, n-\|\tilde{c}(g)\|_1\}\Big\}.$$

This number can be calculated in $\mathcal{O}(m+n)$ time when $\mathrm{inv}(g)$ is given.

Let $(r', c')$ be the parity pair of a coset leader. We remark that the coset leader is non-unique when $\|r'\|_1 > 1$ or $\|c'\|_1 > 1$. Denote

$$\alpha = \max\{\|r'\|_1, \|c'\|_1\} \quad \text{and} \quad \beta = \min\{\|r'\|_1, \|c'\|_1\}.$$

To count the number of coset leaders, we recall the idea of the construction that we have used. The first phase matches each 1 in $r'$ to a distinct 1 in $c'$ to minimize the number of lit lights. There are totally $\binom{\alpha}{\beta}\beta!$ combinations. If $\|r'\|_1 = \|c'\|_1$, then the construction is done, and the number of combinations $\binom{\alpha}{\beta}\beta! = \|r'\|_1! = \|c'\|_1!$ is achieved. Otherwise, we proceed to the second phase, described as follows:

The remaining unmatched 1's must either be only in the row parity vector or only in the column parity vector. We group them two by two so that if we put each group in the same row/column, the row/column parity will not be toggled. If $\|r'\|_1 > \|c'\|_1$, then each group can choose any of the $n$ columns. If $\|c'\|_1 > \|r'\|_1$, then each group can choose any of the $m$ rows. By applying a similar technique to generate the Lagrange coefficients for the Lagrange interpolation, we can combine the two cases into one formula. That is, each group has $\left(\frac{\alpha-\|c'\|_1}{\|r'\|_1-\|c'\|_1}n + \frac{\alpha-\|r'\|_1}{\|c'\|_1-\|r'\|_1}m\right)$ choices. It is not hard to see that we have enumerated all possibilities that can achieve the minimal number of lit lights. The number of groups is

$$\binom{\alpha-\beta}{2}\binom{\alpha-\beta-2}{2}\cdots\binom{2}{2} = (\alpha-\beta)!\,2^{\frac{\beta-\alpha}{2}}.$$

Therefore, the number of possible coset leaders when $\|r'\|_1 \neq \|c'\|_1$ is

$$\begin{aligned}&\binom{\alpha}{\beta}\beta!(\alpha-\beta)!\,2^{\frac{\beta-\alpha}{2}}\left(\frac{\alpha-\|c'\|_1}{\|r'\|_1-\|c'\|_1}n + \frac{\alpha-\|r'\|_1}{\|c'\|_1-\|r'\|_1}m\right)\\ =\,&\alpha!\,2^{\frac{\beta-\alpha}{2}}\left(\frac{\alpha-\|c'\|_1}{\|r'\|_1-\|c'\|_1}n + \frac{\alpha-\|r'\|_1}{\|c'\|_1-\|r'\|_1}m\right).\end{aligned}$$

Combining the two cases, the number of possible coset leaders is

$$\begin{cases}\|r'\|_1! & \text{if } \|r'\|_1 = \|c'\|_1,\\ \binom{\alpha}{\beta}\beta!(\alpha-\beta)!\,2^{\frac{\beta-\alpha}{2}}\left(\frac{\alpha-\|c'\|_1}{\|r'\|_1-\|c'\|_1}n + \frac{\alpha-\|r'\|_1}{\|c'\|_1-\|r'\|_1}m\right) & \text{otherwise.}\end{cases}$$

### *4.3. Covering Radius*

The covering radius problem of the game asks for the maximal number of lights remained among all games when the player minimizes the number of lights. With the understanding of cosets, the desired number is represented by

$$\max_{C\in G/S}\min_{g\in C}\|g\|_1.$$

**Even-by-Even Games**. All even-by-even games are solvable, so the covering radius is trivially 0.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



**Even-by-Odd Games**. Recall that for each coset $C$,

$$\min_{g\in C}\|g\|_1 = \min\{\|\tilde{r}(g)\|_1, m - \|\tilde{r}(g)\|_1\}.$$

As the syndrome is a vector in $\mathbb{F}_2^{(m-1)\times 1}$, and each possible vector in $\mathbb{F}_2^{(m-1)\times 1}$ corresponds to a distinct coset, we can calculate the covering radius of even-by-odd games as follows:

$$\max_{C\in G/S}\min_{g\in C}\|g\|_1 = \max_{b\in\{0,1,\ldots,m-1\}}\min\{b, m-b\} = \frac{m}{2}.$$

**Odd-by-Odd Games**. Consider a syndrome $(\tilde{r}, \tilde{c})$ of a game in a coset $C$. Let $b = \|\tilde{r}\|_1$ and $a = \|\tilde{c}\|_1$. Each possible syndrome in $\mathbb{F}_2^{(m-1)\times 1} \times \mathbb{F}_2^{1\times(n-1)}$ corresponds to a distinct coset, therefore it suffices to consider all possible $b \in \{0, 1, \ldots, m-1\}$ and $a \in \{0, 1, \ldots, n-1\}$. Note that we are considering the syndrome but not the parity pair, therefore $a - b$ can either be even or odd.

Case I: $a - b$ is even. Then, we have

$$\min_{g\in C}\|g\|_1 = \min\{\max\{b, a\}, \max\{m-b, n-a\}\}.$$

To maximize this number among all cosets, we need to check both $(b, a) = (0, n-1)$ and $(b, a) = (m-1, 0)$, which gives $\min\{n-1, m\}$ and $\min\{m-1, n\}$ respectively. That is, if we restrict ourselves to those $a - b$ that are even, the covering radius is

$$\max\{\min\{n-1, m\}, \min\{m-1, n\}\} = \min\{m, n\} - \delta_{m,n},$$

where $\delta_{m,n}$ is the Kronecker delta, i.e.,

$$\delta_{m,n} = \begin{cases} 1 & \text{if } m = n, \\ 0 & \text{otherwise.} \end{cases}$$

Case II: $a - b$ is odd. Then, we have

$$\min_{g\in C}\|g\|_1 = \min\{\max\{b, n-a\}, \max\{m-b, a\}\}.$$

To maximize the number among all cosets, we need to check both $(b, a) = (1, 0)$ and $(b, a) = (0, 1)$, which gives $\min\{n, m-1\}$ and $\min\{n-1, m\}$ respectively. That is, if we restrict ourselves to those $a - b$ that are odd, the covering radius is $\max\{\min\{n, m-1\}, \min\{n-1, m\}\}$, which is the same as when $a - b$ is even.

Combining these two cases, the covering radius of odd-by-odd games can be represented as:

$$\max_{C\in G/S}\min_{g\in C}\|g\|_1 = \min\{m, n\} - \delta_{m,n}.$$

### *4.4. Error Correction*

The error correction problem is to find a solvable game by toggling the fewest number of lights individually. Depending on the capability of error correcting, such a corrected game may not be unique.

**Even-by-Even Games**. All even-by-even games are solvable, therefore there is no unsolvable game for discussion.

**Even-by-Odd Games**. When $n > 1$, the Hamming distance is 2, thus the code has single error detecting ability, but with no error correcting ability.

When $n = 1$, the Hamming distance is an even number. Denote such even number as $m$, thus the code has $(m-1)$ error detecting ability and $\lfloor\frac{m-1}{2}\rfloor = \frac{m}{2} - 1$ error correcting ability. Note that an $m \times 1$ game only consists of two codewords, namely $\mathbf{0}_{m,1}$ and $\mathbb{1}_{m,1}$.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



Consider an arbitrary vector $w \in \mathbb{F}_2^{m\times 1}$. If there are $\frac{m}{2}$ number of 0's and $\frac{m}{2}$ number of 1's in $w$, then we cannot correct the error because the Hamming distance from $w$ to either of the codewords is the same. If the number of 0's in $w$ is more than that of 1's, then $w$ is closer to $\mathbf{0}_{m,1}$. If the number of 1's in $w$ is more than that of 0's, then $w$ is closer to $\mathbb{1}_{m,1}$. In the latter two cases, the number of errors is at most $\frac{m}{2} - 1$, thus the closest solvable game is unique. In fact, this code is known as a *repetition code* in coding theory.

**Odd-by-Odd Games**. When $m > 1$ and $n = 1$, the code is again a repetition code as aforementioned. However, in this context, $m$ is an odd number, therefore the code has $\frac{m-1}{2}$ error correcting ability.

Consider an arbitrary vector $w \in \mathbb{F}_2^{m\times 1}$. It is impossible to have the same number of 0's and 1's in $w$. Therefore, we can always correct $w$ to the closest solvable game and such solvable game is unique. A similar result holds for $m = 1$ and $n > 1$ based on symmetry. However, when $m = n = 1$, the Hamming distance is 1, thus the code has no error detecting nor error correcting abilities.

The non-trivial cases are the odd-by-odd games when $m, n > 1$. Suppose we have an unsolvable odd-by-odd game $g \in G \setminus S$. We first calculate the syndrome $\mathrm{inv}(g)$ to identify the coset that the game $g$ belongs to. Then, we find an arbitrary coset leader $e$ of such coset. Recall that the coset leader of a coset is the game that has the minimal number of lit lights. Therefore, the game $g - e$ is a closest solvable game.

The code distance of $3 \times 3$ game is 3, thus it has 2 error detecting abilities and 1 error correcting ability. For games of any other size, the distance is 4, therefore the code has 3 error detecting abilities and 1 error correcting ability. This also implies that for any unsolvable odd-by-odd game $g \in G \setminus S$, the closest solvable game is unique if and only if the coset leader of the coset where the game belongs to has one and only one lit light.

In principle, we can use the syndrome of the game to directly locate the error when there is only 1 error. The procedure is as follows:

1. Calculate the syndrome $\mathrm{inv}(g) = ((\tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\intercal, (\tilde{c}_2, \tilde{c}_3, \ldots, \tilde{c}_n))$ of the given game $g$.
2. Initialize $r = (0, \tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\intercal$, $c = (0, \tilde{c}_2, \tilde{c}_3, \ldots, \tilde{c}_n)$, $A = \|r\|_1$, and $B = \|c\|_1$.
3. If $A - B$ is odd, then let $A = m - \|r\|_1$ and flip all the bits in $r$.
4. If $\max\{A, B\} = 0$ or $\max\{m - A, n - B\} = 0$, then the game $g$ is already solvable, and this procedure is then terminated.
5. If $\max\{A, B\} > 1$ and $\max\{m - A, n - B\} > 1$, then there is more than one error, which implicates that the errors cannot be uniquely corrected. Therefore, the procedure can again be terminated.
6. If $\max\{A, B\} > \max\{m - A, n - B\}$, then flip all the bits in both vectors $r$ and $c$.
7. Let $y$ and $x$ be the indices where the $y$-th entry in $r$ is 1 and the $x$-th entry in $c$ is 1. Then, the $(y, x)$-th light in $g$ is the only error position.

The idea of the above procedure is similar to the construction of the coset leader. We first find a valid parity pair of $g$, i.e., $A - B$ is even. Next, we filter out the cases where the game has no errors, or has more than one error. If $1 = \max\{A, B\} \leq \max\{m - A, n - B\}$, then the current parity pair has exactly one 1 in each of the row and column parity vectors. Otherwise, we have $\max\{m - A, n - B\} = 1$, then we flip both the row and column parity vectors such that there is exactly one 1 in both parity vectors. As a result, the coset leader has exactly one 1 in this case, where the location of the 1 is indicated by the parity pair.

## 5. As an Error Correction Code

In this section, we discuss the use of the solvable game space of a two-state Alien Tiles game as an error correction code. Although the general linear code technique based on the generator matrix and the parity check matrix also work, the corresponding matrices have a complicated form. Here, we describe a natural and simple way for encoding and decoding. We also discuss whether the code is an optimal linear code or not, where optimality means that the linear code has the maximal number of codewords among all possible linear

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



codes under the same set of parameters, for example, field size, codeword length and code distance.

### 5.1. Even-by-Odd Games

We first discuss the $m \times 1$ games. An $m \times 1$ game only consists of two codewords, namely $\mathbf{0}_{m,1}$ and $\mathbb{1}_{m,1}$, so it is simply a repetition code. The message we can encode is either the bit 0 or 1. To encode, we simply map 0 to $\mathbf{0}_{m,1}$ and map 1 to $\mathbb{1}_{m,1}$. The way to decode was discussed in the last section, which is precisely described as follows: Let $a, b \in \{0, 1\}$, where $a \neq b$. If the number of $a$'s in a to-be-decoded word is more than that of $b$'s, then we decode to $a$. If their numbers are equal, then we cannot uniquely decode the word.

Now, consider $n > 1$. All these games have single error detecting ability but no error correcting abilities, because the code distance is 2. As the size of the solvable game space, i.e., the number of codewords, is $|S| = 2^{mn-m+1}$, we have $mn - (mn - m + 1) = m - 1$ bits acting as the redundancy for error detection. The message we can encode is an $(mn - m + 1)$-bits string. Denote the message by $(b_1, b_2, \ldots, b_{mn-m+1})$, a natural way to encode the message is as follows: We first consider an $m \times (n - 1)$ matrix

$$D = \begin{pmatrix} b_1 & b_2 & \cdots & b_{n-1} \\ b_n & b_{n+1} & \cdots & b_{2n-2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{mn-m-n+2} & b_{mn-m-n+3} & \cdots & b_{mn-m} \end{pmatrix}$$

and calculate its row parity vector $r = (r_1, r_2, \ldots, r_m)^\mathsf{T}$. If $b_{mn-m+1} = r_1$, then we construct the game

$$(r \mid D).$$

Otherwise, we construct the game

$$(r + \mathbb{1}_{m,1} \mid D).$$

The row parity vector of the former game is $\mathbf{0}_{m,1}$. The one of the latter game is $\mathbb{1}_{m,1}$. That is, both games are solvable, thus we have encoded the message.

Note that the $(1, 1)$-st entry of the encoded message is the same as $b_{mn-m+1}$. Therefore, we can perform the following steps to decode a game $g$:

1. Calculate the row parity vector $r(g)$ of the game $g$.
2. If $r(g) \neq \mathbf{0}_{m,1}$ and $r(g) \neq \mathbb{1}_{m,1}$, then we have detected the existence of errors, but we cannot decode the game, thus the decoding procedure can be terminated.
3. The game has no errors, thus the message can be recovered by reading the matrix.

Lastly, we discuss the optimality of the code.

**Theorem 5.** *For even-by-odd games, only the $2 \times n$ games and the $m \times 1$ games are optimal linear codes.*

**Proof.** We first consider $m = 2$ and $n > 1$. Recall that $|S| = 2^{2n-1}$. By applying the Singleton bound [26], we have

$$A_2(2n, 2) \leq 2^{2n-2+1} = 2^{2n-1},$$

where $A_q(\ell, d)$ denotes the maximal number of codewords among all possible linear codes with field size $q$, codeword length $\ell$, and code distance $d$. As $A_2(2n, 2) = |S|$, we know that all $2 \times n$ games are optimal linear codes.

Now, consider the case of $m > 2$ and $n > 1$. If there exists a linear code having the same set of parameters, but of more number of codewords, we can conclude that the error correction code (ironically without error correction ability) formed by even-by-odd games

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



(with $n > 1$) is not an optimal linear code. The existence of such code is shown by the following construction. Consider $D = \mathbb{F}_2^{mn-1}$, where $mn-1$ is odd. Take any two arbitrary $a, b \in D$ such that their Hamming distance is 1. One of them has odd number of bits and the other has even number of bits. Without loss of generality, assume $a$ has odd number of bits. We append a parity bit, i.e., 1, at the end of $a$, and append a parity bit, i.e., 0, at the end of $b$. The extended words (vectors) are elements in $\mathbb{F}_2^{mn}$, and their Hamming distance is 2. That is, by appending a parity bit to every word in $D$, we obtain a linear code that has the same set of parameters as the even-by-odd games when $n > 1$. The number of codewords is $2^{mn-1}$, which is greater than $|S| = 2^{mn-m+1}$. This proves the non-optimality of even-by-odd games, with $m > 2$ and $n > 1$.

Lastly, we discuss the case of all $m \times 1$ games. An $m \times 1$ game only consists of two codewords, namely $\mathbf{0}_{m,1}$ and $\mathbb{1}_{m,1}$, so it is simply a repetition code. By applying the Singleton bound [26], we have

$$A_2(m, m) \leq 2^{m-m+1} = 2.$$

As the equality case is attainable, we know that all $m \times 1$ games are optimal linear codes. □

### 5.2. Odd-by-Odd Games

When $m = 1$ or $n = 1$ (but not both), the code is an repetition code. The code is optimal as suggested by the Singleton bound. The way to encode and decode is similar to that for even-by-1 games, except that the numbers of 0's and 1's are never the same. If $m = n = 1$, then there is no unsolvable game, thus we do not consider this case.

In the remaining discussion of this sub-section, we only consider odd-by-odd games, with $m, n \geq 3$.

**Theorem 6.** *For odd-by-odd games (where $m, n \geq 3$), only the $3 \times 3$ games form an optimal linear code.*

**Proof.** See Section E. □

Recall that for all $3 \times 3$ games, the Hamming distance of this code is 3, thus it is 2 error detecting and 1 error correcting. For games of larger sizes, i.e., $m, n \geq 3$ but not $m = n = 3$, their Hamming distances are all 4, thus they are 3 error detecting and 1 error correcting. That is, we can only correct up to one error. Regardless of the optimality, we now discuss the natural way to encode and decode.

The dimension of the solvable game space is $mn - m - n + 2$, so the message we can encode is an $(mn - m - n + 2)$-bits string. Denote the message by $(b_1, b_2, \ldots, b_{mn-m-n+2})$. First, we put the message into a matrix in the form

$$\left(\begin{array}{c|c} b_{mn-m-n+2} & \mathbf{0}_{1,n-1} \\ \hline \mathbf{0}_{m-1,1} & D \end{array}\right) \quad \text{where} \quad D := \begin{pmatrix} b_1 & b_2 & \cdots & b_{n-1} \\ b_n & b_{n+1} & \cdots & b_{2n-2} \\ \vdots & \vdots & \ddots & \vdots \\ b_{mn-m-2n+3} & b_{mn-m-2n+4} & \cdots & b_{mn-m-n+1} \end{pmatrix}.$$

Let $(r, c)$ be the parity pair of $D$. By Theorem 3, the number of bits in $r$ and $c$ must either be both odd or both even. If the number of bits in $r$ (or $c$) is even, then the game

$$\left(\begin{array}{c|c} 0 & c \\ \hline r & D \end{array}\right)$$

has a parity pair $(\mathbf{0}_{m,1}, \mathbf{0}_{1,n})$. If the number of bits in $r$ (or $c$) is odd. Then, the game

$$\left(\begin{array}{c|c} 1 & c \\ \hline r & D \end{array}\right)$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



has a parity pair $(\mathbf{0}_{m,1}, \mathbf{0}_{1,n})$. In both of the above games, if the $(1,1)$-st entry is not equal to $b_{mn-m-n+2}$, then we click the $(1,1)$-st light so that all the lights in the first row and the first column are toggled. Combining these two cases, Table 1 shows the condition to flip all the bits in the parity pair.

**Table 1.** The condition to flip all the bits in the parity pair $(r, c)$ for encoding a message to an odd-by-odd game, where $m, n \geq 3$.

| $\|r\|_1 \bmod 2$ **(or** $\|c\|_1 \bmod 2$**)** | $b_{mn-m-n+2}$ | **Flip?** |
|---|---|---|
| 0 | 0 | No |
| 0 | 1 | Yes |
| 1 | 0 | Yes |
| 1 | 1 | No |

The table is actually the truth table of exclusive OR (XOR). As addition in binary field is equivalent to modulo 2 addition, we can write the desired codeword as

$$\left(\begin{array}{c|c} b_{mn-m-n+2} & c + (\|c\|_1 + b_{mn-m-n+2})\mathbb{1}_{1,n-1} \\ \hline r + (\|r\|_1 + b_{mn-m-n+2})\mathbb{1}_{m-1,1} & D \end{array}\right).$$

To decode a game, we can run the procedure described in Section 4.4. The procedure has three types of output.

- The first type of output indicates the game is a solvable one, so we can extract the message $(b_1, b_2, \ldots, b_{mn-m-n+2})$ directly.
- The second type of output indicates the game has more than one error, so the error correction is non-unique, thus we consider the code as non-decodable.
- The third type of output indicates the location of the single error. Let $(y, x)$ be the location of the error, we can correct the error by flipping the $(y, x)$-th bit of the game grid.

### *5.3. Example*

To demonstrate the error correction code, we provide an example in this sub-section. Consider the $3 \times 3$ games. Suppose we want to encode the message $(1, 1, 0, 1, 0)$. First, the matrix $D$ that consists of the first four bits of the message is

$$D = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}.$$

We first put the parity pair of $D$, which is $(r, c) = ((0, 1)^\intercal, (1, 0))$ to a $3 \times 3$ game:

$$\begin{pmatrix} 0 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \end{pmatrix}.$$

As $\|r\|_1 \bmod 2 = 1$ and the last bit of the message is 0, according to Table 1, we need to flip the first row and the first column except the $(1,1)$-st entry. Thus, the encoded codeword is

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Now, we introduce an error to the codeword. Note that the error can be located at any entry in the codeword, including those bits that are not the message bits. For example, the erroneous codeword is

$$w = \begin{pmatrix} 0 & 0 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{pmatrix}.$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



To correct such error, we first calculate the syndrome of this game, which is $((1,0)^\intercal,(1,1))$ Next, we initialize $r=(0,1,0)^\intercal$ and $c=(0,1,1)$. As $\|r\|_1-\|c\|_1=-1$ is odd, we flip the bits in $r$, thus $r$ becomes $(1,0,1)^\intercal$. Now, we have

$$2=\max\{\|r\|_1,\|c\|_1\}>\max\{3-\|r\|_1,3-\|c\|_1\}=1,$$

so we flip both $r$ and $c$. That is, we have $r=(0,1,0)^\intercal$ and $c=(1,0,0)$. This pair of bits indicates that the error is located at

$$e=\begin{pmatrix}0&0&0\\1&0&0\\0&0&0\end{pmatrix}.$$

Afterwards, we can correct the error by $w-e$ and obtain the codeword

$$\begin{pmatrix}0&0&1\\1&1&1\\0&0&1\end{pmatrix}.$$

By directly reading this matrix, the decoded message $(1,1,0,1,0)$ can be recovered.

**6. States Decomposition**

One may wonder whether we can practically decompose the 4 states of the ordinary (4-state) Alien Tiles to obtain two 2-state Alien Tiles games. We describe a states decomposition method below, where the number of states and the toggle patterns can be arbitrary. Suppose there are $q$ states. We reuse $\Psi$ to denote the matrix of all vectorized toggle patterns. Suppose $q$ has $P$ distinct prime factors. By the fundamental theorem of arithmetic, we can express $q$ in canonical representation

$$q=\prod_{i=1}^{P}p_i^{r_i}$$

where $p_i$ are distinct primes.

To show whether the (vectorized) game $g$ is solvable, it is equivalent to show whether the linear system

$$\Psi k\equiv g\pmod{q}$$

is solvable. By applying the Chinese remainder theorem, we can decompose the system into

$$\begin{cases}\Psi k\equiv g\pmod{p_1^{r_1}}\\ \Psi k\equiv g\pmod{p_2^{r_2}}\\ \quad\vdots\\ \Psi k\equiv g\pmod{p_P^{r_P}}\end{cases}$$

and combine the solutions of the above system into a unique modulo $q$ solution afterwards. Our task becomes solving the linear system in the form of

$$\Psi k\equiv g\pmod{p^r}\tag{1}$$

where $p$ is a prime. If $r=1$, we cannot further decompose the system, which means that we have to directly solve the problem.

Consider $r\geq 2$. The solution of Eq. (1) is also a solution of the linear system

$$\Psi k\equiv g\pmod{p}.$$

Let $k_0$ be a solution of this system, we have

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



$$\Psi k_0 - g = pz$$

for some integer vector $z$ (after vectorization). Let $k_0 + py$ be a solution of Eq. (1), we have

$$\begin{aligned} \Psi(k_0 + py) - g &\equiv 0 \pmod{p^r} \\ (\Psi k_0 - g) + p\Psi y &\equiv 0 \pmod{p^r} \\ pz + p\Psi y &\equiv 0 \pmod{p^r}. \end{aligned}$$

By dividing $p$, we obtain

$$\Psi y \equiv -z \pmod{p^{r-1}}.$$

Repeatedly applying this procedure, we can reduce the power of the prime in the modulo to until $r = 1$.

The main issue of this decomposition is that the solution $k_0$ may not be unique. We need to try which $k_0 + u$ is solvable after we reduce the prime power, where $\Psi u \bmod p$ is equivalent to a zero game. A wrong choice can lead to an unsolvable system. This also happens when we reduce from the 4-state Alien Tiles to the 2-state ones.

However, if $m$ and $n$ are both even, all 4-state Alien Tiles games are solvable [4,21]. In other words, the move sequence to solve an arbitrary game is unique (up to permutation). We do not need to concern the choice of $k_0$ as every even-by-even 2-state Alien Tiles is solvable. A summary of the procedures of solving a 4-state Alien Tiles $g$ by states decomposition is as follows.

1. Find a move sequence $k$ that solves the 2-state Alien Tiles $g$ mod 2.
2. Calculate $z = \frac{1}{2}(\Psi k - g)$ without performing modulo.
3. Find a move sequence $y$ that solves the 2-state Alien Tiles $z$ mod 2.
4. The move sequence in solving the 4-state Alien Tiles $g$ is $k + 2y$.

As an example, consider the 4-state Alien Tiles

$$g = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

where the four states are mapped to 0, 1, 2 and 3 respectively. The first step is to solve the 2-state game

$$g \bmod 2 = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

The move sequence is

$$k = \begin{pmatrix} 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Next, we calculate

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



$$z = \frac{1}{2}(\Psi k - g) = \frac{1}{2}\left(\begin{pmatrix} 2 & 0 & 2 & 2 \\ 0 & 3 & 0 & 0 \\ 0 & 3 & 0 & 0 \\ 2 & 0 & 2 & 2 \end{pmatrix} - \begin{pmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 2 & 0 \\ 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}\right) = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & -1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}.$$

After that, we solve another 2-state game

$$z \bmod 2 = \begin{pmatrix} 1 & 0 & 1 & 1 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 \end{pmatrix}.$$

The move sequence is

$$y = \begin{pmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}.$$

Finally, we obtain the move sequence that solves the 4-state game $g$, which is

$$k + 2y = \begin{pmatrix} 0 & 0 & 2 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 2 & 3 & 1 \\ 0 & 0 & 2 & 0 \end{pmatrix}.$$

## 7. Conclusions

In this paper, we investigated the properties of Alien Tiles, including the solvability, invariants, etc based on different settings. We also discussed the efficient methods to deal with coding-theoretical problems for the game, such as the Hamming weight, the coset leader, and the covering radius. We also demonstrated how to apply the game as an error correction code with a natural way to encode and decode, and verified that particular game sizes can form optimal linear codes. An open problem that remains in this paper is the existence of an efficient method to find out the minimal number of moves (the shortest solution) in solving a solvable game. Lastly, we discussed the states decomposition method and left an open problem on the issue of kernel selection.

One future direction is to study the coding-theoretical problems on the ordinary Lights Out and also its other variants. Different variants have different structures to be explored. Some variants may have easy solutions, e.g., the two-state Alien Tiles that we have discussed in this paper; while some of them may not, e.g., the Gale-Berlekamp switching game. Besides constructing easy solutions, proving how hard a game problem is will also be a potential research direction.

Another future direction is to investigate more structural properties of the ordinary four-state Alien Tiles. In the four-state version, there is a subtle issue when we model the problem via the use of linear algebra. Take the odd-by-odd games as an example: We can click all the lights twice in a) two arbitrary distinct columns, b) two arbitrary distinct rows, or c) an arbitrary column, followed by an arbitrary row, to keep an arbitrary game unchanged. The combination of these actions forms a kernel. As each light must be clicked for even number of times, the kernel is a free module instead of a vector space over $\mathbb{F}_4$, which means that it is a module that has a basis. This is because multiplication in this finite field is not having the same effect as taking modulo after integer multiplication. In particular, applying two double clicks is equivalent to "no clicking", however $2 \times 2 \neq 0$ in $\mathbb{F}_4$.

On top, we have discussed that the states decomposition has an issue on finding a proper element in the kernel that leads to a solvable game in the lower state. However, this also means that we have reduced the search space, by eliminating those elements in the kernel that lead to an unsolvable game in the lower state. A future direction is to investigate whether this decomposition can help to solve constraint programming problems like that described in [27]. This problem is a recreational mathematics problem on Alien Tiles aiming to find the solvable game which has the longest shortest solution. Existing approaches reduce the search space mainly by symmetry breaking [28–30]. The use of our states decomposition together with symmetry breaking in obtaining desired solvable games (which may not be unique) is a potential research direction.

**Author Contributions:** Conceptualization, H.H.F.Y.; methodology, H.H.F.Y., K.H.N. and S.K.M.; validation, H.H.F.Y., H.W.H.W. and H.W.L.M.; formal analysis, H.H.F.Y., K.H.N. and S.K.M.; investigation, H.H.F.Y., K.H.N. and S.K.M.; writing—original draft preparation, H.H.F.Y.; writing—review and editing, H.W.H.W. and H.W.L.M.; visualization, H.H.F.Y.; project administration, H.H.F.Y.; funding acquisition, H.H.F.Y., K.H.N. and H.W.L.M.; All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Data Availability Statement:** Not applicable.

**Acknowledgments:** Most technical details in this paper were included in the final year project of H. H. F. Yin for his B.Eng. degree in 2014 when he was with both the Department of Information Engineering and the Department of Mathematics, The Chinese University of Hong Kong, Shatin, New Territories, Hong Kong. We thank Raymond W. Yeung for allowing H. H. F. Yin to explore freely in his final year project. We also thank Lap Chi Lau and Javad B. Ebrahimi for their discussion on the Gale-Berlekamp switching game. Lastly, we thank Dr. Linqi Guo for his insightful discussion.

**Conflicts of Interest:** The authors declare no conflict of interest.

## Abbreviations

The following abbreviations are used in this manuscript:

| | |
|---|---|
| P | polynomial time |
| NP | non-deterministic polynomial time |
| XOR | exclusive OR |
| BSC | binary symmetric channel |
| ECC | error correction code |
| MDS | maximum distance separable |

## Appendix A. Introduction to Algorithmic Complexity

It takes time for a computer to conduct every operation, e.g., addition, multiplication, etc. An *algorithm* is a finite sequence of operations (which can be directly computed by the computer) to solve a problem or perform a computation. The non-constant parameters of the problem are the inputs of the algorithm, where the input size is measured in bits (the number of binary symbols). The number of operations needed to solve a problem by an algorithm can be expressed as a function of the input size. That is, this function can be regarded as a measure of the time required in solving the problem, with the use of such algorithm. When the input size is significantly large, the running time, which is dominated by the dominant term of the function, to solve the same problem by two different algorithms can be significantly differed.

In this appendix, we roughly describe the expression of time complexity in big O notation, and the complexity classes P, NP, NP-complete and NP-hard. For a more rigorous discussion, we refer readers to computer science textbooks such as [31,32].

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



*Appendix A.1. Complexity Class P*

To describe the efficiency of an algorithm, *big O notation* is used in the context of algorithmic complexity. Let $n$ be the input size, $f(n)$ be the number of operations needed to solve the problem by the algorithm, and $g(n)$ be the comparison function for estimating $f(n)$. We write

$$f(n) = \mathcal{O}(g(n))$$

if there exist positive integers $c$ and $n_0$ such that $f(n) \leq cg(n)$ for all $n \geq n_0$. Then, we say that the algorithm solves the problem in $\mathcal{O}(g(n))$ time. For example, let $f(n) = 5n^3 - 4n^2 + 3n - 2$. As $n$ is a positive integer, we have $f(n) \leq 5n^3 + 4n^3 + 3n^3 + 2n^3 = 14n^3$. Therefore, we know that $f(n) = \mathcal{O}(n^3)$, where the dominant term $n^3$ is extracted by the big O notation.

If the algorithm runs in $\mathcal{O}(g(n))$ time, where $g(n)$ is a finite-degree polynomial expression in $n$, then we say that the algorithm is a *polynomial-time* algorithm. Roughly speaking, the complexity class P consists of all *decision problems*, i.e., yes-no questions, which can be solved by some polynomial-time algorithms. A more rigorous definition involves the definition of deterministic Turing machine, which is out of the scope of our discussion in this paper. As a rule of thumb, the problems in P are considered as "efficiently solvable problems".

*Appendix A.2. Complexity Class NP*

The computer that we considered in the above context can only run operations one by one. That is, when we want to compute multiple branches, we can only compute them in a step-wise manner. A figurative example is that we are solving a maze - at a certain point, we need to determine whether turn left or turn right. In principle, we will need to try both ways to ensure that we can find an exit. That is, we have to come back to this position and try the other way later. Yet, if we have a multicore processor, we can try both ways at the same time in different cores. Suppose the processor has infinitely many cores, and we can solve a problem in polynomial time (in parallel), then we say that the problem can be solved by an algorithm in *non-deterministic polynomial time*. Roughly speaking, the complexity class NP is the collection of all decision problems that can be solved in non-deterministic polynomial time. Similarly, a more rigorous definition involves the non-deterministic Turing machine, but in this context, we omit such an in-depth discussion.

We can regard NP as a class of decision problems that can be verified in polynomial time. For example, suppose we obtain a path in a maze, and we want to verify whether this path could lead to the desired exit. What we need to do is to follow this path and see if we can really reach the exit. This can be done in polynomial time, because this path was generated by one of the cores in polynomial time when solving the maze. Note that P is a subclass of NP, because we can verify the answer in polynomial time by solving the problem as well.

*Appendix A.3. Complexity Class NP-Complete*

In mathematics, we usually transform a problem to another one that we know how to solve. This is known as *reduction*. When we reduce a problem A to a problem B, it means that problem B covers all the instances of problem A. If the reduction can be done in polynomial time, we say that A is reduced to B in polynomial time. In other words, if we can efficiently solve problem B, then we can also efficiently solve problem A with a polynomial-time reduction. However, the converse is not always true. Although it is called "reduction", we are actually transforming a problem to a "harder" problem.

The hardest type of problems in NP is called *NP-complete* problems. That is, any problem in NP can be reduced to an NP-complete problem in polynomial time. If we can solve any of the NP-complete problems in polynomial time, then we can solve all NP problems in polynomial time, which implies that P equals NP. However, the existence of

 doi:10.20944/preprints202206.0331.v1



such a polynomial-time algorithm is still unknown. Proving or disproving the existence of such algorithm can actually solve the "P versus NP problem", which is one of the Millennium Prize Problems in Mathematics [33]. The answer of this problem has crucial consequences and implications in different branches of mathematics and computer science [34,35]. Throughout recent decades, it is widely believed that no such algorithm exists [36–38], but it is yet to be proven.

*Appendix A.4. Complexity Class NP-Hard*

A problem that is reduced from an NP-complete problem in polynomial time is called an *NP-hard* problem. In other words, an NP-hard problem is no "easier" than any NP-complete problems. Note that an NP-hard problem may not be in NP, i.e., we may not be able to verify the answer in polynomial time. If an NP-hard problem is in NP, then the problem is NP-complete, and this is actually a standard technique to prove the NP-completeness of a particular problem. A convention is that the class "NP-hard" is not restricted to decision problems. If the decision problem (e.g., does a solution exist?) is NP-complete, then its associated function problem, i.e., finding an instance of the solution, is NP-hard. This is because if we obtain a solution, we can also answer the decision problem, thus the function problem is no easier than NP-complete.

## Appendix B. Crash Course on Coding Theory

We focus on describing the background of coding theory that is related to this paper. As a convention, when we say coding theory, we refer to *channel coding theory*, which is a study of error correction. We refer the readers to textbooks such as [39,40] for a more comprehensive discussion of coding theory.

*Appendix B.1. Error Correction Codes*

One of the main goals in coding theory is to send a message (represented by a sequence of symbols) from a source to a destination via a "noisy" medium that could introduce "errors" to the message. A common form of "errors" is to modify some symbols in the message. To ensure that the destination can accurately and effectively recover the message, we need to add redundant information to the message to form a *codeword* of an *error correction code*. Due to some practical considerations, such as the way to distinguish different codewords, most error correction codes, e.g., [41,42], have assumed that every codeword is of the same length. Every possible sequence that has the same length as a codeword is called a *word*.

The overall picture is that the source transforms a message $m$ into a codeword $c$, then transmits $c$ to the destination. The destination receives a word $\hat{c}$, which may be different from $c$ due to errors. Then, the destination tries to guess the correct codeword $c$, then transforms it back to the message. It is easy to visualize that some constraints must be imposed on the number of errors, otherwise there is no clue to guess the correct codeword in a systematic and scientific manner.

In convention, we use *Hamming distance* as the metric to measure the number of errors.

**Definition A1** (Hamming Distance)**.** *The* Hamming distance *between two sequences of the same length is defined as the number of symbols at which the corresponding values are different.*

Let $n$ be the length of any word, and $q \geq 2$ be the size of the alphabet used by the words. That is, there are totally $q^n$ possible words. For each possible codeword $c$, let $\mathrm{Ball}_r(c)$ be the set that contains every word $w$ such that the Hamming distance between $w$ and $c$ is at most $r$. In other words, $\mathrm{Ball}_r(c)$ is the closed *Hamming ball* of radius $r$ centered at $c$. By direct counting, the volume of $\mathrm{Ball}_r(c)$ is

$$\mathrm{Vol}(\mathrm{Ball}_r(c)) := \sum_{i=0}^{r} \binom{n}{i} (q-1)^i.$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



**Definition A2** (Distance of Codes). *The* distance *of an error correction code, denoted by $d$, is defined as the minimum Hamming distance between any pair of distinct codewords.*

As long as the Hamming balls $\text{Ball}_r(c)$ of all possible codewords $c$ do not overlap, when we receive a word $w$, we say that the correct codeword is $c$ if $w \in \text{Ball}_r(c)$. To maximize the number of correctable words, we maximize the radius of the balls as long as they do not overlap with each other. The distance of the code, $d$, is either $2r+1$ or $2r+2$. In other words, such code can guarantee to detect at most $d-1$ errors, and correct at most $\lfloor \frac{d-1}{2} \rfloor$ errors.

The aforementioned decoding approach is called the *minimum distance decoding*, or the *nearest neighbor decoding*, which finds the closest codeword by modifying the fewest number of symbols in the received word. A *binary symmetric channel (BSC)* is a channel model such that every symbol (0 or 1) has the same (crossover) probability to be modified into another symbol. For a BSC with a crossover probability of less than 0.5, the minimum distance decoding process is equivalent to the *maximum likelihood decoding* process, which aims at finding a codeword that has the maximum likelihood.

**Definition A3** (Covering Radius). *The* covering radius *$R$ is the smallest $r$ such that all words are covered by the union of all codeword-centered Hamming balls of radius $r$.*

In other words, any words must have a codeword within $R$ Hamming distance. The covering radius is applied in coding for write-once memories [43], football pool betting [44], etc.

*Appendix B.2. Hamming Bound & Singleton Bound*

Given $q$, $n$ and $d$, the maximal number of codewords among all possible codes is denoted by $A_q(n,d)$, and the code having $A_q(n,d)$ codewords is called an *optimal code*. The method to find the exact value of $A_q(n,d)$ for arbitrary $q$, $n$ and $d$ remains to be an open problem, and is known as the *main coding theory problem*. Nevertheless, many bounds of $A_q(n,d)$ have already been established. Some of the bounds are rexpressed in the form of an inequality, e.g., the Plotkin bound [45] and the Gilbert-Varshamov bound [46,47], while some of them are expressed in the form of an optimization problem, e.g., the linear programming bound [48] and the semidefinite programming bound [49].

One of the earliest bounds is the *Hamming bound* [42]. The idea is straightforward: The total volume of all non-overlapping Hamming balls cannot excess the number of all possible words. We can ensure the non-overlapping constraint when the radius of every Hamming ball centered at a codeword is no larger than $\lfloor \frac{d-1}{2} \rfloor$. Mathematically speaking,

$$A_q(n,d) \leq \frac{q^n}{\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i}(q-1)^i}. \qquad \text{(Hamming Bound)}$$

A code such that the equality case of the Hamming bound holds is called a *perfect code*. That is, in a perfect code, every word is covered by one and only one codeword-centered Hamming ball of radius $\lfloor \frac{d-1}{2} \rfloor$, thus every word can be corrected as a unique codeword. However, Tietäväinen [50] has proven the non-existence of perfect codes over a prime-power alphabet except trivial perfect codes (the code distance is either 1 or $n$), Hamming codes (widely used in error correction code (ECC) memory), and Golay codes (used by the NASA's Voyager [51]).

Another bound that we used in this paper is called the Singleton bound [26]. The idea of this bound is that, if we delete the first $(d-1)$ symbols of every codeword, i.e., the length of every codeword is now $(n-d+1)$, then each pair of altered codewords is still separated by a Hamming distance of at least 1. The number of altered codewords, which is the same as the number of original codewords, is at most $q^{n-d+1}$. Mathematically speaking,

$$A_q(n,d) \leq q^{n-d+1}. \qquad \text{(Singleton Bound)}$$

A class of codes known as the *maximum distance separable (MDS) codes* achieves the equality of the Singleton bound. Reed-Solomon code [41] is a type of MDS code that has been applied in many modern technologies, including CDs, QR codes, etc.

*Appendix B.3. Linear Codes*

A *linear code* is a vector space. Linear codes are of interests in the community of coding theory, as we can apply a wide range of tools in linear algebra. However, this also means that we have restricted the alphabet size to a prime power, as the vector space is defined over a finite field. Further, the number of codewords, i.e., the size of the vector space, must be a power of $q$. Given $q$, $n$ and $d$, the maximal number of codewords among all possible linear codes is denoted by $B_q(n,d)$, and the linear code having $B_q(n,d)$ codewords is called an *optimal linear code*. Note that $B_q(n,d)$ must satisfy

$$B_q(n,d) \leq q^{\lfloor \log_q A_q(n,d) \rfloor}.$$

**Definition A4** (Hamming Weight)**.** *The* Hamming weight *of a codeword $c$, denoted by* $\mathrm{wt}(c)$*, is the Hamming distance between $c$ and the zero vector, i.e., the number of non-zero symbols in $c$. The* Hamming weight *of a code is the minimum Hamming weight among all the non-zero codewords of this code.*

Let $u$ and $v$ be two codewords such that the distance in between is $d$. As a vector space, the linear combination of codewords is also a codeword. Therefore, $u - v$ is also a codeword. Note that $\mathrm{wt}(u - v) = d$. In other words, the distance of a linear code equals to the Hamming weight of the code. Despite the simplified criteria, finding the Hamming weight of a code is an NP-hard problem [16].

In the following, we express the message $m$ and the codeword $c$ as row vectors. To encode $m$, we calculate $c = mG$, where $G$ is called the *generator matrix* of the linear code. The generator matrix is associated with a *parity check matrix* $H$ such that $GH^\intercal$ gives a zero matrix. In standard form, $G$ is written as a block matrix $(I \mid P)$ for some matrix $P$ and identity matrix $I$. Thus, we can write $H = (-P^\intercal \mid I)$. Note that the identity matrices in $G$ and $H$ may be of different dimensions.

Although encoding is easy, decoding is generally an NP-hard problem [7,8], unless we can exploit some special structures of the code, e.g., as described in [52]. For linear codes, we can extract some information regarding the error pattern. In particular, let $w$ be the received word. We can express $w := c + e$ for some error $e$ from a codeword $c$. Recall that $GH^\intercal$ gives a zero matrix, therefore we have

$$wH^\intercal = (c+e)H^\intercal = mGH^\intercal + eH^\intercal = eH^\intercal,$$

which is known as the *syndrome* of $w$, and such syndrome identifies the error pattern. In other words, if we pre-compute the syndromes for all possible correctable error patterns, i.e., the syndromes of all $e$ with $\mathrm{wt}(e) \leq \lfloor \frac{d-1}{2} \rfloor$, then we have a lookup table of size $\sum_{i=0}^{\lfloor \frac{d-1}{2} \rfloor} \binom{n}{i}(q-1)^i$ for mapping a syndrome to $e$. We can correct a word $w$ by $w - e$, where $e$ is obtained by looking up the syndrome of $w$ from the table, i.e., $wH^\intercal$. This approach is known as *syndrome decoding*.

Further, let $C$ be the abelian group of all codewords under vector addition. As $C$ is a normal subgroup of $\mathbb{F}_q^n$ under vector addition, we have the quotient group $\mathbb{F}_q^n / C = \{C + e \colon e \in \mathbb{F}_q^n\}$. Every entry in the quotient group is a coset.

**Definition A5** (Coset Leader)**.** *The word that has the minimum Hamming weight in each coset is the* coset leader *of this coset.*

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



That is, the coset leader is the minimum-weight error pattern that leads or maps a codeword to a word in this coset. We can view the decoding procedure of a word in the coset as subtracting the given word by the coset leader of the coset.

*Appendix B.4. Repetition Codes, Hamming Codes, and Extended Hamming Codes*

For simplicity, we only consider binary field in this sub-section.

Repetition code is one of the simplest codes that repeats a single bit to be encoded into a length $n$ codeword. That is, there are totally two codewords in this code. To decode a word, we look for the symbol (0 or 1) that dominates the bit string. If there are more 0s than 1s, then we decode to the bit 0, and vice versa. The distance of the code is $n$, therefore every repetition code achieves the equality of the Singleton bound, thus being an optimal code.

Hamming code [42] is a perfect code, i.e., achieves the equality of the Hamming bound, thus it is an optimal code. Let $r \geq 2$. A (binary) Hamming code is a linear code with codeword length $2^r - 1$ and code distance 3, with a parity check matrix whose columns consist of all the non-zero vectors of $\mathbb{F}_2^r$. The dimension of the code (i.e., vector space) is $2^r - r - 1$, hence the number of codewords is $2^{2^r-r-1}$. As the distance is 3, the Hamming code is exactly equivalent to a single error correcting code. An example of the parity check matrix for the case of $r = 3$ is

$$\begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 & 1 \end{pmatrix}.$$

It is remarked that the order of the columns are not fixed. However, if the columns are arranged in the order of increasing binary numbers, i.e., the $j$-th column is the binary representation of the number $j$, then the syndrome can directly indicate the location of the error. More precisely, if the syndrome is the binary representation of the number $j$, then the error is located at the $j$-th bit.

$Extended(binary)Hammingcode$ is a Hamming code by appending a parity check bit. If $H$ is the parity check matrix of a Hamming code, then the corresponding extended Hamming code has a parity check matrix

$$\left(\begin{array}{ccc|c} & & & 0 \\ & H & & \vdots \\ & & & 0 \\ \hline 1 & \cdots & 1 & 1 \end{array}\right).$$

This code is a linear code of codeword length $2^r$ and code distance 4, thus it can correct exactly one error. As the extra bit is only a parity bit, the number of codewords remains the same as the Hamming code. As a result, the dimension of the code is $2^r - r - 1$, and the number of codewords is $2^{2^r-r-1}$.

## Appendix C. Proof of Theorem 2

This appendix outlines the proof of Theorem 2 for even-by-odd games, which states that $\tilde{r}$ is a surjective function that maps a game to its syndrome. In the following, the game space $G$ consists of all $m \times n$ games, where $m$ is even and $n$ is odd.

Recall that $\tilde{r}$ is an invariant function in the case of even-by-odd games, i.e., for any move sequence $k \in K$, we have $\tilde{r}(g) = \tilde{r}(g + \psi(k))$ for all $g \in G$. Further, for any game $g \in C$ with $C \in G/S$, we have $g + \psi(k) \in C$.

If $\tilde{r}$ is surjective, we know that $[G : S] \geq 2^{m-1}$. Suppose we have a set $M$ that consists of $2^{m-1}$ games. If, for every game $g \in G$, there is a move sequence $k \in K$ such that $g + \psi(k) \in M$, then we know that $[G : S] \leq 2^{m-1}$. That is, if both of the above conditions are true, then we have $[G : S] = 2^{m-1}$. Also, due to the property of a surjective map and the invariance properties, we know that $\tilde{r}$ gives a syndrome, and this completes the proof. In the remaining text of this Appendix, we prove the above two conditions.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



First, we consider the following construction. Let $(\bar{r}_2, \bar{r}_3, \ldots, \bar{r}_m)^\intercal \in \mathbb{F}_2^{(m-1)\times 1}$ be an arbitrary binary vector. We construct a game $\bar{g}$ with entries

$$\bar{g}_{ij} = \begin{cases} \bar{r}_i & \text{if } j = 1 \text{ and } i > 1, \\ 0 & \text{otherwise.} \end{cases}$$

That is,

$$\bar{g} = \left(\begin{array}{c|c} 0 & \mathbf{0}_{1,n-1} \\ \hline \bar{r}_2 & \\ \vdots & \mathbf{0}_{m-1,n-1} \\ \bar{r}_m & \end{array}\right).$$

By the definition of $\tilde{r}$, we have $\tilde{r}(\bar{g}) = (\bar{r}_2, \bar{r}_3, \ldots, \bar{r}_m)^\intercal$, thus $\tilde{r}$ is automatically surjective based on the construction.

For the second condition, we construct the set $M$, where $|M| = 2^{m-1}$ as follows:

$$M = \left\{ \left(\begin{array}{c|c} 0 & \mathbf{0}_{1,n-1} \\ \hline v & \mathbf{0}_{m-1,n-1} \end{array}\right) : v \in \mathbb{F}_2^{(m-1)\times 1} \right\}.$$

Consider an arbitrary game $g \in G$. Let $\bar{g}$ be the sub-game formed by removing the first column of $g$. Note that $\bar{g}$ is solvable because it is an even-by-even game. By applying the move sequence that solves $\bar{g}$ on $g$, we obtain a new game $\tilde{g}$, with the 2-nd to the $n$-th columns being all zero vectors. As a result, if $\tilde{g}_{11} = 0$, then $\tilde{g} \in M$.

Suppose $\tilde{g}_{11} = 1$, we apply the move sequence to flip the first column (i.e., click every light in the first column and the first row on, except the $(1,1)$-st light) of $\tilde{g}$, as a result, the $(1,1)$-st light is switched off. That is, after the above moves, the game is in $M$. This completes the proof of the second condition, and thus the entire proof of Theorem 2 is completed as well.

## Appendix D. Proof of Theorem 4

This Appendix outlines the proof of Theorem 4 for odd-by-odd games, which states that inv is a surjective function that maps $g$ to its syndrome. Here, the game space $G$ is only applicable for $m \times n$ games, where both $m$ and $n$ are odd.

The idea of the proof is the same as that for even-by-odd games in Section C, but its construction process is a bit different. We first recall the following properties. For any game $g \in G$ and any move sequence $k \in K$, we have $\mathrm{inv}(g) = \mathrm{inv}(g + \psi(k))$. Hence, the game $g$ must fall into one of the cosets in $G/S$. If $g \in C$, where $C \in G/S$, then $g + \psi(k) \in C$.

If inv is surjective, then we have $[G : S] \geq 2^{m+n-2}$. Suppose we have a set $M$ that consists of $2^{m+n-2}$ games. If, for every game $g \in G$, there is a move sequence $k \in K$ such that $g + \psi(k) \in M$, then we have $[G : S] \leq 2^{m+n-2}$. When the above two conditions are both true, we have $[G : S] = 2^{m+n-2}$. Due to the properties of a surjective map and invariance, we conclude that inv gives a syndrome, and the proof is completed. Similarly, we now proceed to the proof of the above two conditions.

Let $\tilde{R} = (\tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\intercal \in \mathbb{F}_2^{(m-1)\times 1}$ and $\tilde{C} = (\tilde{c}_2, \tilde{c}_3, \ldots, \tilde{c}_n) \in \mathbb{F}_2^{1\times(n-1)}$ be two arbitrary binary vectors. We construct two games $A, B \in G$ with entries

$$A_{ij} = \begin{cases} \tilde{r}_i + \sum_{k=2}^{m} \tilde{r}_k & \text{if } j = 1 \text{ and } i > 1, \\ \sum_{k=2}^{m} \tilde{r}_k & \text{if } j = 1 \text{ and } i = 1, \\ 0 & \text{otherwise;} \end{cases} \quad \text{and} \quad B_{ij} = \begin{cases} \tilde{c}_j + \sum_{k=2}^{n} \tilde{c}_k & \text{if } j > 1 \text{ and } i = 1, \\ \sum_{k=2}^{n} \tilde{c}_k & \text{if } j = 1 \text{ and } i = 1, \\ 0 & \text{otherwise.} \end{cases}$$

The summation in the definition of $A_{ij}$ means that if the number of bits in $\tilde{R}$ is odd, we flip all entries of the first column. Similarly, we flip the whole first row of $B_{ij}$. Thus, the number of lit lights in the first column of $A$ and the number of lit lights in the first row of $B$ are both

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



even. The invariants of $A$ and $B$ are then respectively given by $\mathrm{inv}(A) = (\tilde{R}, \mathbf{0}_{1,n-1})$ and $\mathrm{inv}(B) = (\mathbf{0}_{m-1,1}, \tilde{C})$.

To merge these two invariants, consider the game $g = A + B \in G$. There are four cases.

Case I: $A_{11} = B_{11} = 0$. We have $\mathrm{inv}(g) = (\tilde{R}, \tilde{C})$.

Case II: $A_{11} = B_{11} = 1$. Note that $g_{11} = 0$, and the numbers of lit lights in the first row and the first column of $g$ are both odd. Therefore, we have

$$r(g) = (0, \tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\mathsf{T} + \mathbb{1}_{m,1} \quad \text{and} \quad c(g) = (0, \tilde{c}_2, \tilde{c}_3, \ldots, \tilde{c}_n) + \mathbb{1}_{1,n}.$$

As $r_1(g) = c_1(g) = 1$, by the definition of $\tilde{r}$ and $\tilde{c}$, we have $\mathrm{inv}(g) = (\tilde{R}, \tilde{C})$.

Case III: $A_{11} = 1$ and $B_{11} = 0$. Note that $g_{11} = 1$, and the number of lit lights in the first column is even, but that in the first row is odd. Therefore, we have

$$r(g) = (0, \tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\mathsf{T} + \mathbb{1}_{m,1} \quad \text{and} \quad c(g) = (0, \tilde{c}_2, \tilde{c}_3, \ldots, \tilde{c}_n).$$

As $r_1(g) = 1$ and $c_1(g) = 0$, we have $\mathrm{inv}(g) = (\tilde{R}, \tilde{C})$ from the definitions of $\tilde{r}$ and $\tilde{c}$.

Case IV: $A_{11} = 0$ and $B_{11} = 1$. Note that $g_{11} = 1$, which is precisely the symmetry case of Case III. The number of lit lights in the first column is odd, but that in the first row is even. Therefore, we have

$$r(g) = (0, \tilde{r}_2, \tilde{r}_3, \ldots, \tilde{r}_m)^\mathsf{T} \quad \text{and} \quad c(g) = (0, \tilde{c}_2, \tilde{c}_3, \ldots, \tilde{c}_n) + \mathbb{1}_{1,n}.$$

As $r_1(g) = 0$ and $c_1(g) = 1$, we have $\mathrm{inv}(g) = (\tilde{R}, \tilde{C})$ from the definitions of $\tilde{r}$ and $\tilde{c}$.

Combining all above four cases, for any $(\tilde{R}, \tilde{C}) \in \mathbb{F}_2^{(m-1)\times 1} \times \mathbb{F}_2^{1\times(n-1)}$, there exists a game $g \in G$ such that $\mathrm{inv}(g) = (\tilde{R}, \tilde{C})$. This implicates that inv is a surjective map.

We now proceed to prove the second condition. Construct the set $M$, where $|M| = 2^{m+n-2}$ by

$$M = \left\{ \left( \begin{array}{c|c} 0 & w \\ \hline v & \mathbf{0}_{m-1,n-1} \end{array} \right) : v \in \mathbb{F}_2^{(m-1)\times 1}, w \in \mathbb{F}_2^{1\times(n-1)} \right\}.$$

Consider an arbitrary game $g \in G$. By removing the first row and the first column of $g$, we obtain an even-by-even sub-game $\bar{g}$. Being an even-by-even game, $\bar{g}$ is a solvable game. By applying the move sequence that solves $\bar{g}$ on $g$, we obtain another game $\tilde{g}$ of the form

$$\tilde{g} = \left( \begin{array}{c|c} u & w \\ \hline v & \mathbf{0}_{m-1,n-1} \end{array} \right),$$

where $u \in \mathbb{F}_2$, $v \in \mathbb{F}_2^{(m-1)\times 1}$ and $w \in \mathbb{F}_2^{1\times(n-1)}$.

If $u = 0$, then $\tilde{g} \in M$. If not, then we click the $(1,1)$-st light on, so that the first row and the first column are flipped. The resultant game is then in $M$. Therefore, the proof of the second condition is done, and Theorem 4 is proved as well.

## Appendix E. Proof of Theorem 6

We first show the optimality of the linear code formed by the $3 \times 3$ games. Note that the Hamming distance of this code is 3, therefore, we first apply the Hamming bound and obtain

$$A_2(9,3) \leq \frac{2^9}{\binom{9}{0} + \binom{9}{1}} = 51.2.$$

As the number of codewords of a linear code (the size of a vector space) must be a power of the size of the field, we have

$$B_2(9,3) \leq 2^{\lfloor \log_2 51.2 \rfloor} = 32.$$

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 24 June 2022 doi:10.20944/preprints202206.0331.v1



The size of the solvable game space, i.e., the number of codewords, is $|S| = 2^{mn-m-n+2} = 2^5 = 32$, which means that the upper bound of $B_2(9,3)$ is attainable. This also implicates that the error correction code formed by $3 \times 3$ games is an optimal linear code.

For other games of larger sizes, i.e., $m, n \geq 3$, but not both $m = n = 3$, their Hamming distances are all 4. Therefore, the number of codewords is $|S| = 2^{mn-m-n+2}$. We can show that this code is not optimal by showing the existence of a linear code that has more than $|S|$ codewords, where the codeword length is $mn$ and the code distance is 4. The existence of such a linear code is the result of *shortening* an extended (binary) Hamming code. Shortening means that for any linear code of codeword length $\ell$, dimension $k$ (i.e., the number of codewords is $q^k$) and code distance $d$, there exists a linear code of codeword length $\ell - x$, dimension $k - x$ and code distance $d$ for any $1 \leq x \leq k - 1$. It is one of the propagation rules of modifying linear codes, which can be achieved by removing some columns of the parity check matrix. The proof of shortening can be found in standard textbooks on Coding Theory such as [40].

Recall that an extended Hamming code is a linear code of codeword length $2^r$, dimension $2^r - r - 1$ and code distance 4, where $r \geq 2$ is an integer. Our goal is to show that there exist integers $r \geq 2$ and $t \geq 0$ such that $2^r - t = mn$ and $2^r - r - 1 - t > mn - m - n + 2$ are satisfied.

We proceed as follows: first show that there exist integers $r \geq 2$ and $t \geq 0$ satisfying $2^r - t = mn$ and $2^r - r - 1 - t = mn - m - n + 2$. In other words, we want to show that we can shorten an extended Hamming code to obtain a linear code that has the same set of parameters as that of the code formed by the game. To satisfy both conditions, we have $2^r - mn = 2^r - r - 1 - mn + m + n - 2$, which gives $r = m + n - 3 > 2$. Furthermore, $t = 2^{m+n-3} - mn$. We now show that $t \geq 0$ by induction.

Due to symmetry, we only consider the parameter $m$ in the induction process. The base case is either $m = 3$, $n = 5$ or $m = 5$, $n = 3$. That is, $t = 2^5 - 15 = 17 \geq 0$. Assume that $2^{m+n-3} - mn \geq 0$ for some $m$ and $n$. For the case of $m + 2$, we have

$$\begin{aligned} & 2^{(m+2)+n-3} - (m+2)n \\ &= (2^{m+n-3} - mn) + (2^{m+n-3} - 2n) + 2^{m+n-2} \\ &> 2(2^{m+n-3} - mn) + 2^{m+n-2} \geq 0. \end{aligned}$$

Therefore, the propositions $r = m + n - 3 > 2$ and $t = 2^{m+n-3} - mn \geq 0$ are verified by induction arguments.

Next, we show that there exist integers $2 \leq r' < m + n - 3$ and $t' \geq 0$ satisfying $2^{r'} - t' = mn$ and $2^{r'} - r' - 1 - t' > mn - m - n + 2$. In particular, we let $r' = r - 1 \geq 2$. Then,

$$mn = 2^r - t = 2^{r'} - (t - 2^{r'}) = 2^{r'} - t',$$

where $t' = t - 2^{r'} = 2^{m+n-4} - mn$. We adopt a similar induction approach to show that $t' \geq 0$. The base case is $t = 2^4 - 15 = 1 \geq 0$. Applying the induction step on the case of $m + 2$, we have $2^{(m+2)+n-4} - (m+2)n > 2(2^{m+n-4} - mn) + 2^{m+n-1} \geq 0$. Now, consider

$$2^r - r - 1 - t = (2^{r'} - r' - t') - 2 = mn - m - n + 2.$$

Therefore, we have $2^{r'} - r' - r' = mn - m - n + 4 > mn - m - n + 2$. This shows that we can shorten an extended Hamming code to obtain a linear code that possesses the same codeword length and code distance as the code formed by the game, however this linear code has more than $|S|$ codewords. In other words, the code formed by the game is not optimal.

## References

1. Kreh, M. "Lights Out" and Variants. *The American Mathematical Monthly* **2017**, *124*, 937–950.

2. The Mathematics of Lights Out. https://www.jaapsch.net/puzzles/lomath.htm.
3. Anderson, M.; Feil, T. Turning Lights Out with Linear Algebra. *Mathematics Magazine* **1998**, *71*, 300–303.
4. Rhoads, G.C. A Group Theoretic Solution to the Alien Tiles Puzzle. *Journal of Recreational Mathematics* **2007**, *36*, 92–103.
5. Goldwasser, J.; Klostermeyer, W.; Trapp, G. Characterizing Switch-Setting Problems. *Linear and Multilinear algebra* **1997**, *43*, 121–135.
6. Sutner, K. The $\sigma$-Game and Cellular Automata. *The American Mathematical Monthly* **1990**, *97*, 24–34.
7. Berlekamp, E.; McEliece, R.; van Tilborg, H. On the Inherent Intractability of Certain Coding Problems. *IEEE Transactions on Information Theory* **1978**, *24*, 384–386.
8. Barg, S. Some New NP-Complete Coding Problems. *Problemy Peredachi Informatsii* **1994**, *30*, 23–28.
9. Stern, J. Approximating the Number of Error Locations within a Constant Ratio is NP-Complete. In Proceedings of the 10th International Symposium on Applied Algebra, Algebraic Algorithms and Error-Correcting Codes, 1993, pp. 325—-331.
10. Arora, S.; Babai, L.; Stern, J.; Sweedyk, Z. The Hardness of Approximate Optima in Lattices, Codes, and Systems of Linear Equations. *Journal of Computer and System Sciences* **1997**, *54*, 317–331.
11. Roth, R.M.; Viswanathan, K. On the Hardness of Decoding the Gale-Berlekamp Code. *IEEE Transactions on Information Theory* **2008**, *54*, 1050–1060.
12. Karpinski, M.; Schudy, W. Linear Time Approximation Schemes for the Gale-Berlekamp Game and Related Minimization Problems. In Proceedings of the Proceedings of the Forty-First Annual ACM Symposium on Theory of Computing, 2009, pp. 313–322.
13. Carlson, J.; Stolarski, D. The Correct Solution to Berlekamp's Switching Game. *Discrete Mathematics* **2004**, *287*, 145–150.
14. Schauz, U. Colorings and Nowhere-Zero Flows of Graphs in Terms of Berlekamp's Switching Game. *The Electronic Journal of Combinatorics* **2011**, *18*. Art. no. P65.
15. Brualdi, R.A.; Meyer, S.A. A Gale-Berlekamp Permutation-Switching Problem. *European Journal of Combinatorics* **2015**, *44*, 43–56.
16. Vardy, A. Algorithmic Complexity in Coding Theory and the Minimum Distance Problem. In Proceedings of the Proceedings of the Twenty-Ninth Annual ACM Symposium on Theory of Computing, 1997, pp. 92–109.
17. Pickover, C.; Mckechnie, C. Alien Tiles Official Web Page. http://www.alientiles.com/.
18. Lights Out Variant: Flipping the Whole Row and Column. https://math.stackexchange.com/questions/441571/lights-out-variant-flipping-the-whole-row-and-column.
19. Strategy for Modified Lights Out. https://puzzling.stackexchange.com/questions/58374/strategy-for-modified-lights-out.
20. The Pilots Brothers' Refrigerator. http://poj.org/problem?id=2965.
21. Maier, P.; Nickel, W. Attainable Patterns in Alien Tiles. *The American Mathematical Monthly* **2007**, *114*, 1–13.
22. Torrence, B. The Easiest Lights Out Games. *The College Mathematics Journal* **2011**, *42*, 361–372.
23. Characterize the Nullspace of a Given Matrix in $\mathbb{F}_2^{n\times n}$. https://math.stackexchange.com/questions/1056944/characterize-the-nullspace-of-a-given-matrix-in-mathbbf-2n-times-n.
24. Minimal Number of Moves Needed to Solve a "Lights Out" Variant. https://math.stackexchange.com/questions/1052609/minimal-number-of-moves-needed-to-solve-a-lights-out-variant.
25. How can I Further Optimize This Solver of a Variant of "Lights Out"? https://stackoverflow.com/questions/27436275/how-can-i-further-optimize-this-solver-of-a-variant-of-lights-out.
26. Singleton, R. Maximum Distance $q$-nary Codes. *IEEE Transactions on Information Theory* **1964**, *10*, 116–118.
27. 027: Alien Tiles Problem. https://www.csplib.org/Problems/prob027/.
28. Gent, I.; Linton, S.; Smith, B. Symmetry Breaking in the Alien Tiles Puzzle. Technical Report APES-22-2000, APES Research Group, 2000.
29. Gent, I.P.; Harvey, W.; Kelsey, T. Groups and Constraints: Symmetry Breaking during Search. In Proceedings of the International Conference on Principles and Practice of Constraint Programming, 2002, pp. 415–430.
30. McDonald, I.; Smith, B. Partial Symmetry Breaking. In Proceedings of the International Conference on Principles and Practice of Constraint Programming, 2002, pp. 431–445.
31. Cormen, T.H.; Leiserson, C.E.; Rivest, R.L.; Stein, C. *Introduction to Algorithms*, 3 ed.; MIT Press, 2009.
32. Wegener, I. *Complexity Theory: Exploring the Limits of Efficient Algorithms*; Springer Science & Business Media, 2005.
33. Jaffe, A.M. The Millennium Grand Challenge in Mathematics. *Notices of the AMS* **2006**, *53*, 652–660.
34. Cook, S. The Importance of the P versus NP Question. *Journal of the ACM* **2003**, *50*, 27–29.
35. Fortnow, L. *The Golden Ticket: P, NP, and the Search for the Impossible*; Princeton University Press, 2013.
36. Gasarch, W.I. Guest Column: The P=?NP Poll. *ACM SIGACT News* **2002**, *33*, 34–47.
37. Gasarch, W.I. Guest Column: The Second P=?NP Poll. *ACM SIGACT News* **2012**, *43*, 53–77.
38. Gasarch, W.I. Guest Column: The Third P=?NP Poll. *ACM SIGACT News* **2019**, *50*, 38–59.
39. Huffman, W.C.; Pless, V. *Fundamentals of Error-Correcting Codes*; Cambridge University Press, 2003.
40. Ling, S.; Xing, C. *Coding Theory: A First Course*; Cambridge University Press, 2004.
41. Reed, I.S.; Solomon, G. Polynomial Codes over Certain Finite Fields. *Journal of the Society for Industrial and Applied Mathematics* **1960**, *8*, 300–304.
42. Hamming, R.W. Error Detecting and Error Correcting Codes. *The Bell System Technical Journal* **1950**, *29*, 147–160.
43. Cohen, G.; Honkala, I.; Litsyn, S.; Lobstein, A. *Covering Codes*; Elsevier, 1997.

44. Hämäläinen, H.; Honkala, I.; Litsyn, S.; Östergård, P. Football Pools —- A Game for Mathematicians. *The American Mathematical Monthly* **1995**, *102*, 579–588.
45. Plotkin, M. Binary Codes with Specified Minimum Distance. *IRE Transactions on Information Theory* **1960**, *6*, 445–450.
46. Gilbert, E.N. A Comparison of Signalling Alphabets. *The Bell system technical journal* **1952**, *31*, 504–522.
47. Varshamov, R.R. Estimate of the Number of Signals in Error Correcting Codes. *Docklady Akademii Nauk SSSR* **1957**, *117*, 739–741.
48. Delsarte, P. An Algebraic Approach to the Association Schemes of Coding Theory. PhD thesis, Université catholique de Louvain, 1973.
49. Schrijver, A. New Code Upper Bounds from the Terwilliger Algebra and Semidefinite Programming. *IEEE Transactions on Information Theory* **2005**, *51*, 2859–2866.
50. Tietäväinen, A. On the Nonexistence of Perfect Codes over Finite Fields. *SIAM Journal on Applied Mathematics* **1973**, *24*, 88–96.
51. Laeser, R.P.; McLaughlin, W.I.; Wolff, D.M. Engineering Voyager 2's Encounter with Uranus. *Scientific American* **1986**, *255*, 36–45.
52. Forney, G. Generalized Minimum Distance Decoding. *IEEE Transactions on Information Theory* **1966**, *12*, 125–131.