Error-Correcting Codes Derived from Cellular Automata Games

Damla Zerya Aslan

July 14, 2024

Introduction

- Cellular Automata games are models where players interact with a grid of cells that evolve over time.
- In two-player Celata games players alternately change the state of cells.
- Error-correcting codes are constructs for detecting and correcting transmission errors.
- Linear error-correcting codes use algebraic methods for encoding and decoding.

Preliminaries

- Games on a digraph $G = (V, E)$ with $V = \{z_1, \ldots, z_n\}$.
- Tokens distributed on vertices; moves involve selecting and moving tokens.
- Vertex labeling: N if the next player has a winning move, P otherwise.

$$F(u) = \{v \in V : (u, v) \in E\}$$

$$u \in P \iff F(u) \subseteq N$$
$$u \in N \iff F(u) \cap P = \emptyset$$

Preliminaries

- The numerical value of a vector $u = (u_0, \ldots, u_{n-1}) \in GF(2)^n$ is:

$$|u| := \sum_{i=0}^{n-1} u_i 2^i$$

- The weight of u is the number of 1-bits in u:

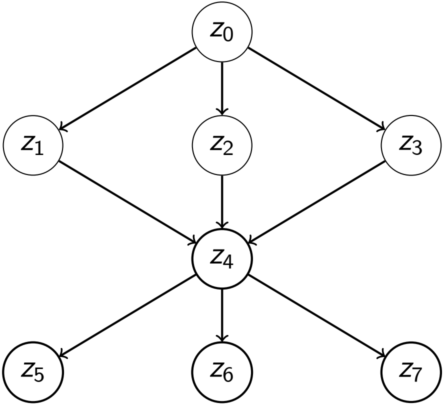
$$w(u) = \sum_{i=0}^{n-1} u_i$$

- The Grundy number (or nimber) $g(x)$ of a position x in a combinatorial game is defined recursively as:

$$g(u) = \text{mex}\{g(v) : v \in F(u)\}$$

Example

- Consider the game graph below and compute Grundy numbers:



$$g(z_0) = 1$$
$$g(z_1) = 0$$
$$g(z_2) = 0$$
$$g(z_3) = 0$$
$$g(z_4) = 1$$
$$g(z_5) = 0$$
$$g(z_6) = 0$$
$$g(z_7) = 0$$

Vector Matrix

- Construct the vector matrix W for the game:

$$W = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

- Each row corresponds to a vertex z_i in the game.

Vector Space Operations

Theorem

In the vector space V, the sum of any two vertices $u, v \in V$ is given by their vector addition in $GF(2)$.

Proof.

$$u = \sum_{i=0}^{n-1} u_i z_i$$

$$v = \sum_{i=0}^{n-1} v_i z_i$$

$$u \oplus v = \sum_{i=0}^{n-1} (u_i \oplus v_i) z_i$$

$\square$

Vector Space Operations

- Summing vectors in $GF(2)$:

$$\begin{aligned} u &= (1,0,0,0,1,0,0) \\ v &= (0,1,0,1,0,0,0) \\ u \oplus v &= (1,1,0,1,1,0,0) \end{aligned}$$

- Hamming distance between vectors:

$$\begin{aligned} u &= (1,0,0,0,1,0,0) \\ v &= (0,1,0,1,0,0,0) \\ H(u,v) &= w(u \oplus v) = w(1,1,0,1,1,0,0) = 4 \end{aligned}$$

Lexicodes

Definition

Lexicodes are a type of error-correcting code that can be generated using lexicographic order on binary vectors.

- Lexicodes are constructed by selecting vectors in lexicographic order with a minimum Hamming distance.

$$\text{Lexicode } L = \{v \in V_m : \text{Hamming distance } d \geq 5\}$$

$$g(z_i) = \text{mex}\{g(z_{i_1}) \oplus g(z_{i_2}) \oplus \ldots \oplus g(z_{i_j})\}$$

Lexicode Algorithm

- Calculate $g(z_m)$ for each state m.
- Identify seeds (not sums of smaller g-values).
- Generate basis members from seeds.
- Apply greedy algorithm for lexicodes.

$$\text{Lexicode } L = \{v \in V_m : \text{Hamming distance } d \geq 5\}$$

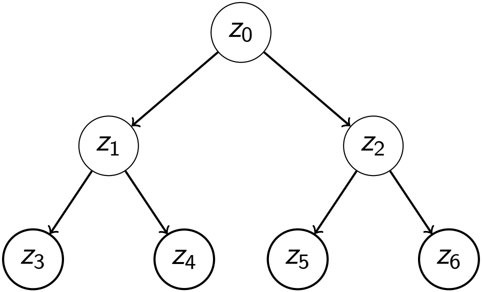
Example

- Consider a game with $n = 5$ and 3 dimensions.
- Basis vectors:

$$v_1 = (1, 0, 0, 0, 0), \quad v_2 = (0, 1, 0, 0, 0), \quad v_3 = (0, 0, 1, 0, 0)$$

- Linear independence means no vector in the set can be represented as a linear combination of the others.

$$a_1 v_1 + a_2 v_2 + a_3 v_3 = 0$$
$$a_1 = a_2 = a_3 = 0$$



Greedy Algorithm for Lexicodes

- Start with the smallest vector not yet in the code.
- Add vectors in lexicographic order, ensuring a minimum Hamming distance.
- Example: Start with $v_1 = (1, 0, 0, 0, 0)$.
- Add $v_2 = (0, 1, 0, 0, 0)$ if $H(v_1, v_2) \geq d$.

  Lexicode $L = \{v \in GF(2)^5$: Hamming distance $d \geq 2\}$

- Construct the vector matrix W for the game:

$$W = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \end{pmatrix}$$

- Each row corresponds to a basis vector.
- Lexicodes:

$$\{(1,0,0,0,0),(0,1,0,0,0),(0,0,1,0,0),(0,0,0,1,0),$$

$$(0{,}0{,}0{,}0{,}1),\ (0{,}1{,}0{,}0{,}1),\ (0{,}0{,}1{,}1{,}0),\ (0{,}0{,}1{,}0{,}1)\}$$

Forcing a Win

- Strategy based on g-function to determine P-, N-, and D-positions.
- Representations and follower functions.

$$\mathrm{Fe}(u_e, u_j, v_j) = u_e \cup \{v_j\} \setminus \{u_j\}$$

$$\mathrm{Fe}(u_e) = \bigcup_{j=1}^{h} \bigcup_{v_j \in F(u_j)} \mathrm{Fe}(u_e, u_j, v_j)$$

Lexicodes

- Apply equations for forcing a win to identify winning positions.
- Use these positions to derive lexicodes restricted to winning configurations.

$$\mathrm{Fe}(u_e, u_j, v_j) = u_e \cup \{v_j\} \setminus \{u_j\}$$

$$\mathrm{Fe}(u_e) = \bigcup_{j=1}^{h} \bigcup_{v_j \in F(u_j)} \mathrm{Fe}(u_e, u_j, v_j)$$

$$\gamma(u_e, \xi(v_e)) = w_e \subseteq R_j, \quad \xi(w_e) \in F(\xi(v_e)) \cap V_p$$

- Construct lexicodes using only winning positions identified by $\mathrm{Fe}(u_e)$ and $\gamma(u_e, \xi(v_e))$.

Thank You for Listening!