GRAY CODES AND PUZZLES ON ITERATED COMPLETE GRAPHS

ELIZABETH WEAVER

ADVISOR: PAUL CULL
OREGON STATE UNIVERSITY

ABSTRACT. There exists an infinite family of puzzles, the SF Puzzle, that correspond to labelings on iterated complete graphs of odd dimensions, the simplest being the well-known Towers of Hanoi puzzle. There is also a puzzle, known as Spin-Out that corresponds to the reflected binary Gray code; however, thus far there has been no family of puzzles defined that can be represented by labelings on all iterated complete graphs of even dimensions. This paper explores the properties that labelings of known puzzles possess, tries to determine the necessary labeling properties for a new puzzle, and defines a puzzle on K_4^n. Unfortunately, this puzzle does not extend to labelings on all even dimension iterated complete graphs.

1. INTRODUCTION

Mathematics forms the basis for many well-known and interesting puzzles. For example, the popular game the Towers of Hanoi can be represented graphically and has several applications in graph theory and in error correcting codes [2]. By mathematically generalizing the rules of the Towers of Hanoi, a new family of puzzles was created that can be easily represented by graphs [5]. In these puzzles there are an odd number of possible positions for each piece and an easily defined rule to determine the possible moves for each piece. It seems logical that a family of puzzles should exist where there are an even number d possible moves for each piece, but with the exception of $d = 2$ no one rule seems to describe a puzzle like that of the Towers of Hanoi. For $d = 2$, the puzzle has been embodied as Spin-Out, which is available at your local toy shop.

1.1. Definitions From Graph Theory.

Definition 1.1. *A **graph** G consists of a nonempty finite set $V(G)$ of elements called **vertices** and a finite set $E(G)$ of distinct unordered pairs of distinct elements of $V(G)$ called **edges**. Two vertices are **adjacent** if the edge $(v_i, v_j) \in E(G)$. A **subgraph** S of G consists of $V(S) \subset V(G)$ together with the edges connecting any adjacent vertices, $v_i \in V(S)$, $v_j \in V(S)$, and $(v_i, v_j) \in E(G)$.*

Example 1.2. *Figure 1 shows a graph G with vertex set $V(G) = \{a,b,c,d,e\}$ and edge set $E(G) = \{(a,c),(b,d),(b,e),(c,d)\}$ and a subgraph S of G with vertex set $V(S) = \{b,c,d\}$ and edge set $E(S) = \{(b,d),(c,d)\}$.*

Definition 1.3. *The **degree** of a vertex v is equal to the number of vertices adjacent to v.*

Example 1.4. *In Figure 1 the degree of vertex a is 1 and the degree of vertex d is 2.*

Date: August 11, 2005.

This work was done during the Summer 2005 REU program in Mathematics at Oregon State University.

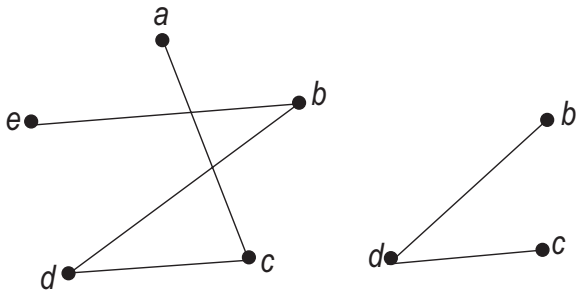

FIGURE 1. Examples of graphs.

Definition 1.5. *A **complete graph on** d **vertices** is a graph in which each pair of distinct vertices are adjacent. It is denoted K_d.*

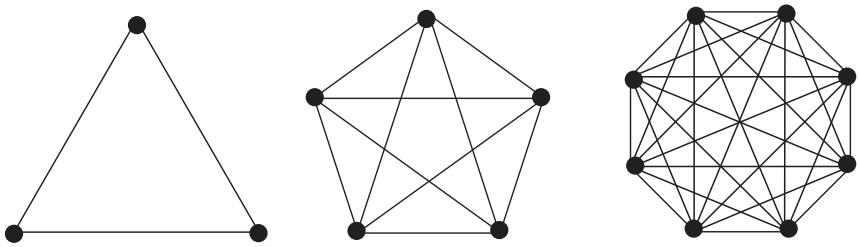

FIGURE 2. The complete graphs K_3, K_5, and K_8.

Definition 1.6. *A **corner vertex** is a vertex of an iterated complete graph K_d^n whose degree is $d-1$.*

Definition 1.7. *An **iterated complete graph** on d vertices with n iterations, denoted K_d^n, can be defined recursively. K_d^1 is the complete graph on d vertices. K_d^n is composed of d copies of K_d^{n-1}. Now, $d-1$ of the corner vertices of each copy are connected to corner vertices of the other $d-1$ copies of K_d^{n-1} so that there is one edge between each pair of copies. Note that this leaves K_d^n with exactly d corner vertices.*

1.2. **Labelings.**

Definition 1.8. *A **labeling scheme** for K_d^n is a method of assigning strings of length n over $\{0,\ldots,d-1\}$ to the vertices of K_d^n which gives a bijection between vertices and strings.*

There are many advantages and disadvantages to most labeling schemes. As a result, there are many different methods for labeling complete iterated graphs. This paper deals mainly with codes that have the Gray Code property, with respect to the Hamming distance between strings.

Definition 1.9. *A labeling of a graph G with the **Gray code property**, with respect to the Hamming distance between strings, is a labeling such that any two adjacent vertices have labels that differ in exactly one position.*

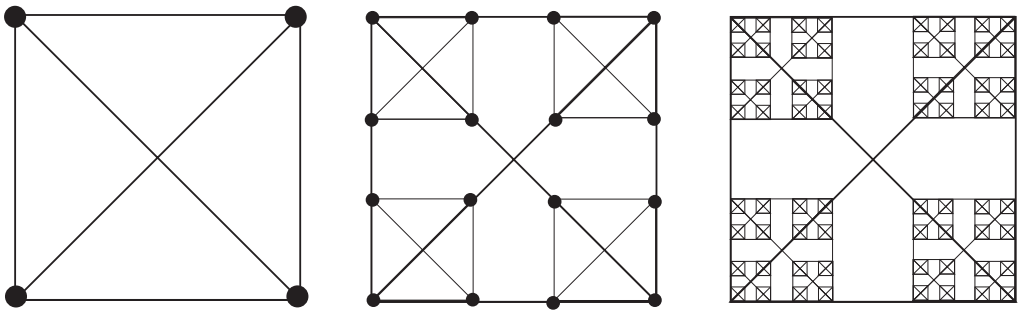

FIGURE 3. The first, second, and fourth iterations of the complete graph on four vertices denoted K_4^1, K_4^2, and K_4^4.

2. THE TOWERS OF HANOI AND THE SF PUZZLE

2.1. **The Towers of Hanoi.** The Towers of Hanoi, a game inspired by a Buddhist legend, was created by Edouard Lucas in 1883. The game consists of three towers and n disks of differing diameters stacked in decreasing order of size from bottom to top. The goal of the game is to move all of the disks from one tower to another, one disk at a time, never placing a larger disk on top of a smaller one. Notice that this is a one player game, and other authors might refer to this as a puzzle since it is not an activity which involves two or more players. Since the literature is inconsistent on this point, I will use the terms game and puzzle interchangeably.

In their paper *Perfect Codes, NP Completeness, and Towers of Hanoi Graphs*, Cull and Nelson present an infinite family of labelled graphs inspired by the Towers of Hanoi puzzle [3]. In these graphs, the vertex labels on K_3^n where $n \geq 0$ correspond to possible configurations of the n disks in the puzzle, and the edges represent possible moves. The labels are strings of length n of ternary digits, $\{0,1,2\}$ identifying the three towers, with each digit from left to right representing the position of each disk from smallest to largest.

The Towers of Hanoi can be solved in a minimum of $2^n - 1$ moves which can be thought of as moving from one corner vertex of K_3^n to another corner vertex. In their paper *Towers of Hanoi and Analysis of Algorithms* [1], Cull and Ecklund outline a recursive algorithm to solve the Towers of Hanoi. To solve the puzzle by moving all of the disks from tower 0 to tower 2, the $n-1$ smaller disks must be moved to tower 1 so the largest disk can be moved to tower 2. This is simply another Towers of Hanoi problem with fewer disks. After the largest disk has been moved, the $n-1$ disks must be moved from tower 1 to tower 2, which again is a smaller Towers of Hanoi problem. Based on these observations, they present the following algorithm.

```
PROCEDURE HANOI(A,B,C,n-1)
    IF n=1 THEN move the top disk from tower A to tower C
           ELSE HANOI (A,C,B,n-1)
                move the top disk from tower A to tower C
                HANOI (B,A,C,n-1)
```

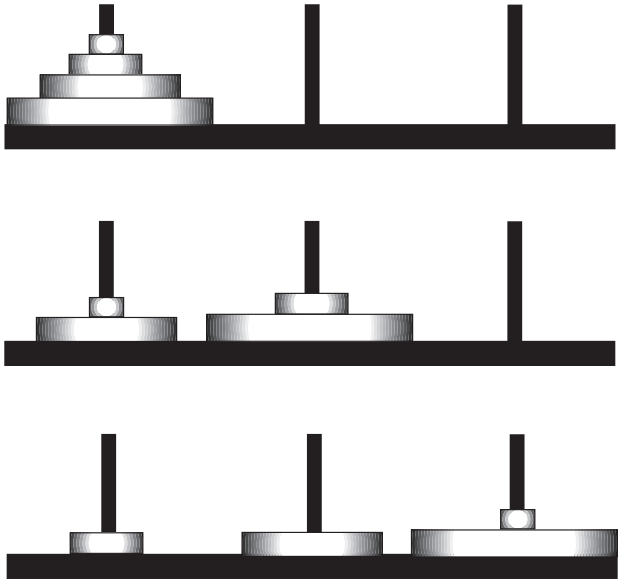

FIGURE 4. The Towers of Hanoi configurations that correspond to the vertex labels 0000, 0101, and 2012 respectively.

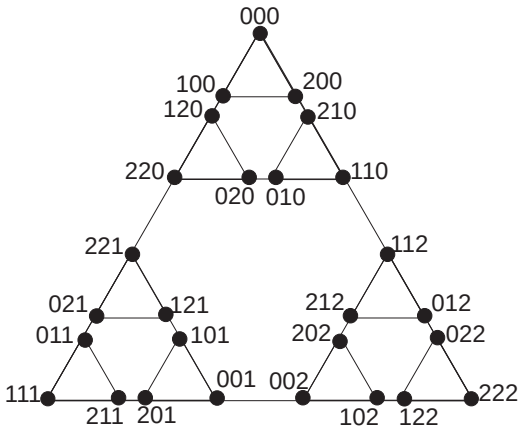

FIGURE 5. The labelled graph corresponding to the Towers of Hanoi with 3 disks.

2.2. **The SF Labeling.** The SF Labeling, created by Stephanie Kleven in 2003 [6], is a labeling scheme that generates a Gray code on iterated complete graphs of odd dimensions, and the SF Labeling of K_3^n is the labelled graph corresponding to the Towers of Hanoi. This labeling can easily be extended to recursively label any K_d^n where d is odd. To explain this labelling, we assume that K_d^n is embedded in the plane.

For $n = 1$ the label of the top vertex of K_d^1 is 0 and the labels of the $d-1$ remaining vertices increase by one counterclockwise around the graph beginning at the top.

To construct the SF Labeling on K_d^n where $n > 1$:

(1) Permute each digit z in the labeling of K_d^{n-1} by α, where $\alpha(z) = (\frac{d+1}{2})z \pmod d$. Then make d copies of the permuted graph.
(2) Index the d copies of the permuted graph from 0 to $d-1$. Rotate the k^{th} copy by $\frac{2\pi k}{d}$ radians clockwise, and add a k to the end of every string in the copy.
(3) Arrange the d copies of K_d^{n-1} so that the top vertex is labelled by a string of n zeros, and the remaining $d-1$ copies are arranged so that their indices increase by one counterclockwise around the graph beginning at the top. Then connect the copies so that each K_d^{n-1} subgraph is connected to every other subgraph by exactly one edge.

The SF Labeling has many desirable properties including an easily defined recursive construction and the Gray code property. The labels of the complete corner vertices of the SF Labeling of K_d^n are strings of n identical digits, and therefore they are easily identifiable. This labeling has applications in the field of error correcting codes, codeword recognizers and error correction machines can easily be constructed for this labeling scheme.

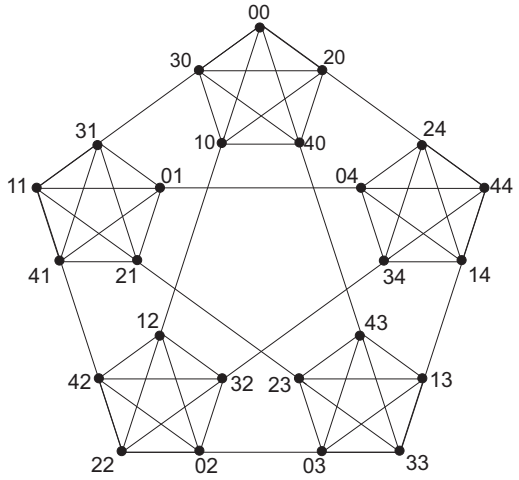

FIGURE 6. The SF Labeling of K_5^2.

2.3. **The SF Puzzle.** Since the SF labeling on K_3^n describes an interesting and elegant puzzle, it is logical that this puzzle can be generalized to correspond with the SF labelings on K_d^n for all odd d. In 2004, Kathleen King introduced a family of puzzles derived from these labelings in her paper *A New Puzzle Based on the SF Labelling of Iterated Complete Graphs*[5]. This new puzzle, called the SF Puzzle, is much like the Towers of Hanoi, but a few rules are added to restrict the possible moves to those represented by the labeling on K_d^n.

The game is played with an odd number of towers and n disks of differing diameters stacked in decreasing order of size from bottom to top. No disk may be placed on top of a smaller disk and no disk may be moved at all unless all of the disks smaller than it are stacked together on another tower. When a disk is able to move to a new tower, the tower that it is allowed to move to is dictated by the disk's current position and the position of the stack of smaller disks. Let the towers be numbered from 0 to $d-1$, the stack of smaller disks be on tower a, and the disk to be moved

be on tower b. Then, the disk may be moved to tower number $2a - b (\mathrm{mod}\ d)$. The vertex labels of the graph correspond to the puzzle in the same way that they do in the Towers of Hanoi graph. The position of the digit indicates the disk, and the value of the digit represents the tower number.

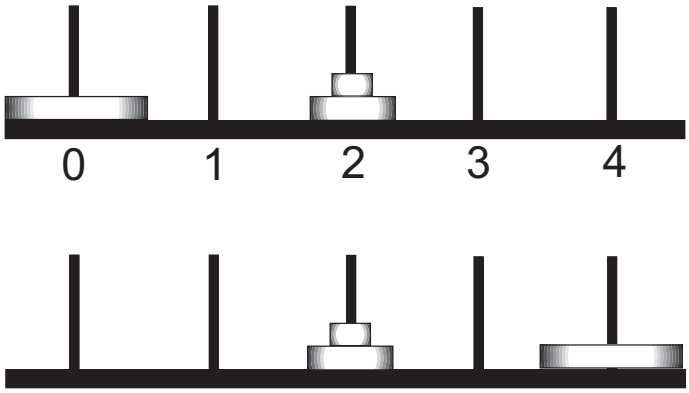

FIGURE 7. The top configuration of disks corresponds to the vertex 220 on the SF labeling of K_5^3. The bottom configuration shows the allowable move for the largest disk in the top picture and corresponds to the label 224.

This family of puzzles can all be solved in a minimum of $2^n - 1$ moves, and King describes an algorithm that efficiently determines the distance between any two configurations (or vertex labels) in O(n) time. This is helpful in determining the number of moves needed to solve the puzzles when arbitrary configurations are chosen as the starting position and the solution position.

3. BINARY CODES AND PUZZLES

3.1. **Reflected Binary Gray Code.** The reflected binary Gray code is a well-known labeling scheme on K_2^n with an easily defined recursive construction [9].

For $n = 1$, let the reflected binary Gray code, G_1, be 0-1. That is, we label one vertex 0 and the other vertex 1.

To construct G_n:

(1) Take two copies of G_{n-1}, and add zeros to the ends of the labels in the first copy and add ones to the ends of the labels in the second copy.
(2) Reverse the order of the strings in the second copy, and connect the new beginning of the second copy to the end of the first copy.

Note that since the first string in the code and the last string in the code differ by exactly one digit, the code is in fact a cycle and can be viewed as a Hamilton cycle in the binary n-cube.

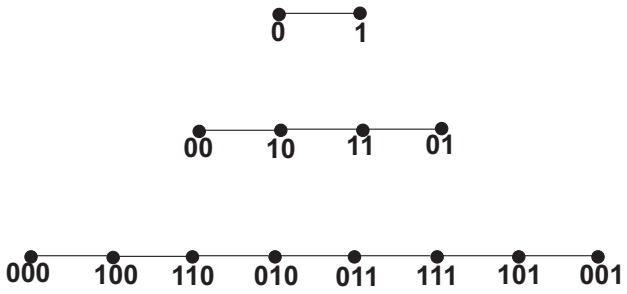

FIGURE 8. The reflected binary Gray code for $n = 1$, $n = 2$, and $n = 3$.

3.2. **Spin Out.** An engaging puzzle that can be represented by a Gray code is the game Spin Out by ThinkFun. The goal of the game is to remove a rectangle with seven spinners on it from a plastic case. In the traditional starting position, all seven spinners are vertical, and the rectangle can only be removed when all of the spinners are aligned horizontally. Let the spinners be labelled 0 to 6 from the left to the right. The n^{th} spinner can only be turned when the spinners 0 through $n-2$ are horizontal and spinner $n-1$ is vertical. Note that the leftmost spinner is free to move at anytime.

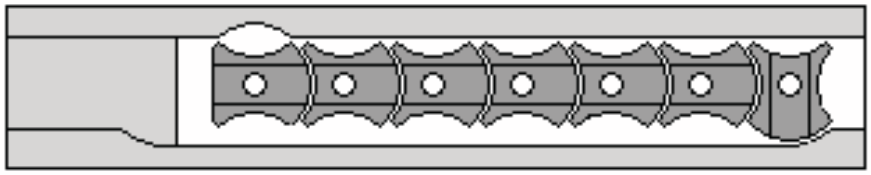

FIGURE 9. The Spin Out game in its starting configuration.

To represent this puzzle by a labeling on a graph, let the horizontal position of a spinner be represented by 0 and the vertical position be represented by 1. Let each configuration of the spinners be represented by a string of seven digits, the leftmost digit corresponding to the leftmost spinner and so on. When the labelled vertices representing spinner configurations are connected by edges representing possible moves, since only one piece moves at a time and there are only two possible positions for each piece, the labeling scheme is a Gray code on K_2^7.

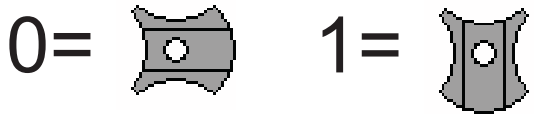

FIGURE 10. The labels corresponding to the piece positions.

Note that the puzzle can be generalized to use any number of spinners. This family of puzzles can be represented by the reflected binary Gray code on K_2^n. To solve the puzzle, or move the n spinners from 11...1 to 11...1, Pruhs presents a two part recursive algorithm [7]. Let "rotate(i)" mean to rotate spinner number i from 1 to 0 or from 0 to 1, where $i \in \{1,...,n\}$, and the spinners are indexed from 1 to n from left to right.

```
PROCEDURE A(n:Integer)
```

Comment: takes puzzle from 1^n to $0^{n-2}10$

```
    IF n=1 THEN "rotate(1)"
           ELSE IF n=2 THEN "rotate(1) rotate(2)"
           ELSE BEGIN
            A(n-2)
            "rotate (n)"
            C(n-2)
            END
END


PROCEDURE C(k:Integer)
```

Comment: takes puzzle from $0^{k-1}1$ to 0^k or from 0^k to $0^{k-1}1$.

```
    IF k=1, THEN "rotate(1)"
            ELSE IF n=2 THEN "rotate(1) rotate(2)"
            ELSE BEGIN
                C(k-1)
                "rotate (k)"
                C(k-1)
                END
END
```

Using these algorithms, it is possible to determine that Spin-Out with n spinners can be solved in $\frac{2^{n+2}-3-(-1)^n}{6}$ moves. An easy way to think of solving this puzzle is to first move the leftmost spinner and repeat this process until the spinners are all horizontally aligned. Also, 00...0 and 00...1 are corner vertices of the reflected binary Gray code on K_2^n which makes it an interesting starting position for the pieces in Spin-Out. Using these spinner configurations, the puzzle can be solved in $2^n - 1$ moves.

4. Properties Of Corner Vertex Labelings

To construct a labeling scheme on K_d^n where d is even that corresponds to a family of puzzles, it is necessary to determine what properties are desirable. Since the previous puzzles mentioned all involve moving only one piece at a time, I determined that the Gray code property should be present. I also wanted the corner vertices to all be of the same form so they would correspond to similar solution configurations as in the SF Puzzle. Also, in the SF Puzzle, all of the disks start on the same tower together and end up together on a different tower. This creates corner vertex labels that consist of digits of identical strings when represented by a labelled graph. However, it is impossible to preserve this property along with the Gray code property in a labeling on K_d^n for all even d and all $n \in \mathbb{N}$. In fact the Spin-Out labeling dose not have this corner labeling property.

4.1. **Corner Vertex Labelings in Binary Gray Codes.**

Definition 4.1. *The **parity** of a binary string is defined to be the sum of its digits mod* 2.

It is important to note that for a Gray code connection to be possible between two vertices their labels must have unequal parity since they are binary strings that differ by a single digit.

Lemma 4.2. *A Gray code labeling with complete corner vertices that are strings of n identical digits on K_2^n is only possible when n is odd.*

Proof. Assume that there exists a Gray code labeling on K_2^n such that the corner vertices are the strings 000...0 and 111...1. Note that the number of vertices in K_2^n is 2^n and therefore is even. To construct a Gray code labeling of K_2^n, strings with parity 0 must connect to strings with parity of 1 and vice versa. Thus the order of the parity of each string in K_2^n is 0101...01. Since the number of vertices is even, the parity of the last string must be 1. Since the corner vertices of K_2^n are 000...0 and 111...1 and the parity of 000...0 is always 0, the parity of 111...1 must equal 1. Therefore n must be odd. □

The following construction, the T-A-B Labeling, creates a Gray code labeling T_n on K_2^n where $n \geq 1$ is odd such that the complete corner vertices are strings of n identical digits.

For $n = 1$ we let T_1 be equal to 0-1. That is, we label one vertex 0 and label the other vertex 1. Clearly this labeling has the desired properties.

When n is even, instead of constructing a Gray code labeling on K_2^n, two codes each made up of 2^n n-bit strings are created. For this construction, call these codes A_n and B_n.

To construct A_n:

(1) Take two copies of K_2^{n-1} and reflect one copy.
(2) Connect the two copies so that the first and last strings are strings of zeros.
(3) Add a zero to the end of every string.

To construct B_n:

(1) Write each string in K_2^{n-1} twice.
(2) Connect the strings so that each string is connected to its copy and the order of the distinct strings in K_2^{n-1} is retained.
(3) Add a one to the end of every string.

To construct a Gray code T_n on K_2^n where n is odd:

(1) Add zeros to the ends of the first 2^{n-2} strings in A_{n-1}, and add ones to the ends of the last 2^{n-2} strings in A_{n-1}.
(2) Index the strings in B_{n-1} by 0, 1, ..., 2^{n-2}. Add a 0 to the ends of the strings with indices equal to 1 or 2 $mod 4$, and add a 1 to the ends of the strings with indices equal to 0 or 3 $mod 4$.
(3) Connect the last string in A_{n-1} to the first string in B_{n-1}.

Proof. When $n = 1$, T_1 clearly has the desired properties. Assume for all odd $n \in \{1, 3, ..., j-2\}$ T_n is a Gray code labeling on K_2^n, and the complete corner vertices are strings of n identical digits. Clearly, because T_{j-2} is a Gray code, the first half of T_j is also a Gray code and contains all j-bit

strings which end in 00 or 01 by construction. The only two places that need to be checked out are at the middle of the sequence and at the end of the sequence. In the middle since T_{j-2} and its reverse are connected, the two labels on either side of the connection agree on $j-2$ characters. When a 0 is added to each of them, they agree on $j-1$ characters. Finally, when the differing last characters are added, they disagree on exactly one character, and the Gray property is preserved.

Now, in the second half of T_j, all strings ending in 11 or 10 are included by construction. Also, the first string in the second half differs from the last string in the first half because each has $j-2$ zeros as its first characters, and 01 is added to one string and 11 is added to the other preserving the Gray property. The Gray property also holds through the second half of T_j because each odd indexed string is identical with it previous even indexed string neighbor and 11 is added to the even indexed string, and 10 is added to the odd indexed string. The string following the odd indexed string differs from it in exactly one of the first $j-2$ digits by the Gray property of T_{j-2}, and both $n-2$-strings have 10 added to them. Finally the last string in T_j consists of j ones because the string of $j-2$ ones is the last string in T_{j-2} and 11 is added to the end. □

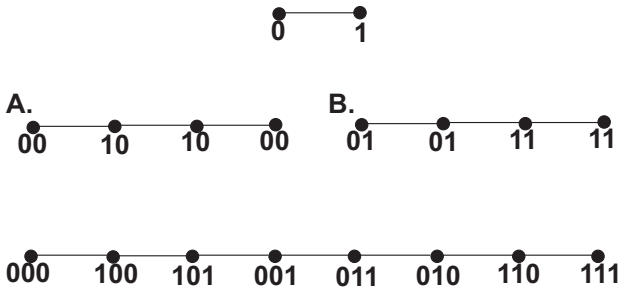

FIGURE 11. The binary Gray codes created by the T-A-B construction for $n = 1$, $n = 2$, and $n = 3$.

Theorem 4.3. *A Gray code labeling with corner vertex labels that are strings of identical digits on K_2^n is possible exactly when n is odd.*

Proof. According to Lemma 4.2 a Gray code labeling with corner vertex labels that are strings of identical digits on K_2^n is only possible when n is odd, and the T-A-B Labeling generates Gray code labelings with corner vertex labels that are strings of identical digits on K_2^n for all odd n. □

4.2. **Properties of Corner Vertex Labelings on Even Dimension Iterated Complete Graphs.**

Theorem 4.4. *It is impossible to have a Gray code labeling on K_d^2 where d is even such that the corner vertex labels are strings of identical digits.*

Proof. Assume that 00...0, 11...1, ..., $(d-1)(d-1)...(d-1)$ are corner vertex labels of a Gray code labeling on K_d^2 where d is even and $d \geq 4$. Now K_d^2 is composed of d K_d^1 subgraphs. Each K_d^1 subgraph is connected to every other K_d^1 subgraph exactly once. Since 00...0, 111...1, ..., $(d-1)(d-1)...(d-1)$ are corner vertex labels they are not connected to any vertices outside of their K_d^1 subgraphs.

Without loss of generality, let two vertices with identical right digits be part of the same K_d^1 subgraph, and $x1$ be a non-corner vertex such that $x \in \{0, 2, ..., d-1\}$. Now there are $d-1$ vertices

outside of the K_d^1 subgraph of $x1$ that have labels that differ from $x1$ by exactly one digit, say $x0$, $x2$, ..., and $x(d-1)$. Since $x \in \{0,2,\ldots,d-1\}$, one of these vertex labels is a string of two identical digits and therefore is a corner vertex. Since this label belongs to a corner vertex and its right digit is not 1, it is not in the same K_d^1 subgraph as $x1$ and is not connected to $x1$.

Now, if $x1$ is connected to one of the remaining $d-2$ labels that differ from it by exactly one digit, there will be an odd number of labels left since d is even. When connections are made between these remaining vertices, there will be a label left with no other label that it can connect to and still preserve Gray code. Therefore, there cannot be a Gray code on K_d^2 with d even, $d \geq 4$, and corner vertex labels that are strings of identical digits. □

Example 4.5. *Assume 00, 11, 22, and 33 are the corner vertex labels of K_4^2. Assume that when two vertex labels have the same right digit they are in the same K_4^1 subgraph. Consider the point 10. It is part of a K_4^1 subgraph and therefore is adjacent to 00, 20, 30. The only other vertices that it can connect to and preserve the Gray code property are 11, 12, and 13. Now 11 is a corner vertex and is only connected to labels whose right digit is 1, so 10 cannot connect to 11. Assume that 11 connects to 12. Now 13 is connected to 03, 23, and 33 in its K_4^1 subgraph, and it can connect to 11, 12, and 10 and preserve the Gray code property. Since 11 is a corner vertex, it cannot connect to 13, and 10 and 12 are already connected to each other. Thus 13 has no possible Gray code connections available to it.*

The idea that I used in the proof of Theorem 4.4 falls aparat when applied to graphs of more than 2 iterations because not every vertex is connected to a corner vertex. I hoped to generalize the idea of parity to apply to bases other than 2 and use that to prove Theorem 4.4 for all n such that n is even, but I was unable to get an argument that supported my hypothesis that the Gray code labeling with corner vertex labels that are strings of identical digits is on possible on odd iterations of even dimension complete graphs.

5. Generalizing Spin Out

5.1. **Piece Restrictions Dictated By The Binary Reflected Gray Code.** In the previous puzzles, there was one piece that was free to move anywhere at any time, while the other pieces all had restrictions placed on their movement. These restrictions proportionally decrease the number of possible moves as the pieces get bigger, or in the case of Spin Out are farther to the right. For example, in the Towers of Hanoi with 3 disks, the smallest disk has 27 possible moves, the middle disk has 9 possible moves, and the largest disk has 3 possible moves.

Consider an extension of Spin Out with n spinners. Label these spinners from 0 to $n-1$ with spinner 0 being the least restricted. Then the digit representing spinner 0 in the Gray code will change from 0 to 1 or from 1 to 0 a total of 2^{n-1} times. The digit representing spinner 1 will change 2^{n-2} times, and spinner $(n-1)$'s digit will change only once.

Instead of writing the Gray code horizontally, I will write it vertically so that each column represents the position of a given piece throughout the puzzle. I will refer to these columns as digit columns throughout this section. Also, consider all of the permutations of the columns of a binary Gray code to be equivalent.

Theorem 5.1. *The reflected binary Gray code is the only binary Gray code where each digit column changes half as many times as the digit column immediately to its left.*

0	0	0
1	0	0
1	1	0
0	1	0
0	1	1
1	1	1
1	0	1
0	0	1

FIGURE 12. The reflected binary Gray code that represents Spin Out with three spinners. The digit columns represent the spinners 0, 1, and 2 from left to right. The changes the spinners make throughout the puzzle are marked in bold.

Proof. For all $n \in \mathbb{N}$, let G_n be a Gray code of n-bit strings. Let the digits be labelled d_0 to d_{n-1} from left to right, and without loss of generality, let 00...0 be the first string. We want to show that for all $n \in \mathbb{N}$, if for all $j \in \{0,...,n-1\}$, d_j changes 2^j throughout G_n, then G_n is a binary reflected Gray code. When $n = 1$, 0-1 is the only code of 1-bit binary strings. The digits in the d_0 digit column change once, and the code is the binary reflected Gray code for 1-bit strings.

To show by induction that this holds for all $n \in \mathbb{N}$, assume if for all $j \in \{0,...,n-1\}$, d_j changes 2^j times throughout G_n, then G_n is a binary reflected Gray code. To show that this is true for $n+1$, assume that for all $j \in \{0,...,n\}$, d_j changes 2^j throughout G_{n+1}. Let $n \geq 2$. Note that since this code contains all of the possible binary strings of $n+1$ bits, each digit column will contain an equal number of ones and zeros. By assumption, d_n changes 2^n times through G_{n+1} which consists of 2^{n+1} strings. Now, if any digit changes two times in a row, the code will not consist of 2^{n+1} distinct strings. Therefore, d_0 must change in every other string, so the digits in the d_0 digit column are 01100...110, and the first 2^n digits of d_0 will consist of 2^{n-1} zeros and 2^{n-1} ones. Also, d_n changes one time in G_{n+1}, and G_{n+1} consists of 2^{n+1} distinct strings, so the bits in the d_n digit column must consist of 2^n zeros and then 2^n ones.

Now d_1 changes two times and consists of 2^n zeros and 2^n ones. If d_1 changed twice in the first 2^n strings, then it would not be possible for it to contain 2^n ones. Thus d_1 changes once in the first 2^n strings and once in the second 2^n strings. Since d_1 changes the same number of times in each half of the code and its digit column consists of half zeros and half ones, each half of the code must also consist of half zeros and half ones. Similarly, all d_j such that $j \in \{0,...,n\}$ change 2^{j-1} times in the first 2^n strings, the number of zeros in the first half of the strings equals the number of zeros in the second half, and the number of ones in the first half of the strings equals the number of ones in the second half.

Note that the number of changes in the first 2^n bits of column d_k such that $k \in \{0,...,n-1\}$ correspond to the number of changes in the column d_k in G_n, and the first 2^n bits of d^k consist of 2^{n-1} zeros and 2^{n-1} ones. Thus the first 2^n bits of the digit columns d_0 through d_{n-1} contain exactly one copy of G_n. Since all of the strings in columns d_0 through d_{n-1} change an even number of times and start with 0, they also all end in zero. Also, since each column d_k contains the same number of zeros in the first and second halves of the code, the same number of ones in the first and

second halves of the code, 2^{k-1} changes in the first and second halves of the codes, and ends with a zero, the second 2^n strings in columns d_0 through d_{n-1} contain a reflection of G_n. By adding d_0, which consists of 2^n zeros and then 2^n ones, to G_n and its reflection a reflected binary code of $(n+1)$-bit strings is created. □

5.2. **A New Labeling On K_4^n.** Since Spin Out can be represented by an easily constructed Gray code and is the only binary Gray code with the desired property to create piece restrictions, it is a logical choice for the simplest puzzle in a new family of puzzles represented by labelings on even dimension iterated complete graphs. Since the corner vertex labels of Spin Out are 00...0 and 00...1, I wanted the corner vertex labels in the new puzzle's labeling scheme to be of a similar form. For example, a puzzle on K_d^n would have corner vertex labels 00...0, 00...1, ..., 00...(d-1).

I will now present a new Reflection-Method labeling scheme on K_4^n that generates a Gray code with corner vertex labels 00...0, 00...1, 00...2, and 00...3 and serves as a new puzzle labeling scheme.

When n=1, the top vertex of K_4^n is labeled 0 and each other vertex, moving counterclockwise, is labeled with the successive integer through 3.

To construct the Reflection-Method labeling scheme on K_4^n:

(1) Construct 4 copies of the Reflection-Method labeling scheme on K_4^{n-1}, and index these copies from 0 to 3.
(2) Reflect copy 1 vertically, and add a 1 to the end of every label in the copy.
(3) Reflect copy 2 both vertically and horizontally, and add a 2 to the end of every label in the copy.
(4) Reflect copy 3 horizontally, and add a 3 to the end of every label in the copy.
(5) Connect the K_4^{n-1} subgraphs so that copy 0 is the top copy and the other copies are connected so that each copy is indexed with the successive integer, moving counterclockwise, ending with $d-1$.

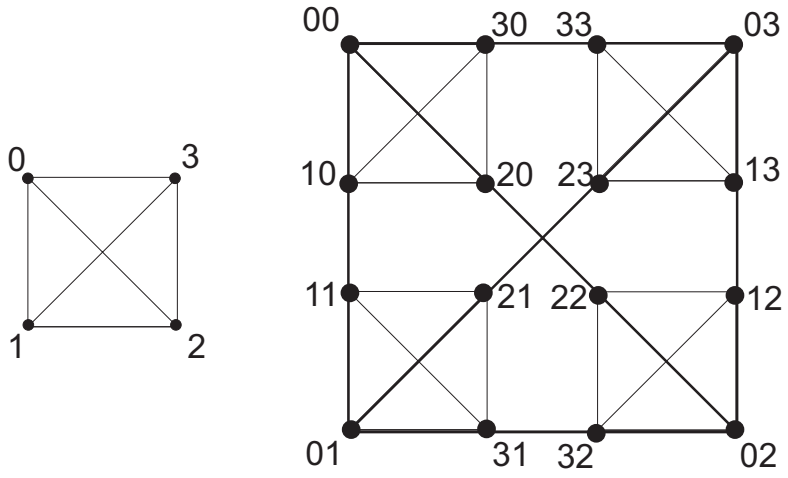

FIGURE 13. The Reflection-Method labelings of K_4^1 and K_4^2.

5.3. **A New Puzzle On K_4^n.** When a modular addition rule similar to that of the SF Puzzle is added to the Reflection-Method labeling scheme, a new family of puzzles is generated. A physical representation for this puzzle has not been created, but moves can be determined by looking only at the vertex labels. The pieces from least restricted to most restricted are represented by the bits in the label string from left to right, and the alphabet of digits used, $\{0,1,2,3\}$, represents the different possible positions for a piece. The goal of the puzzle is to move the pieces from the 00...0 configuration to 00...1, 00...2, or 00...3. Index the digits in a Reflection-Method label from left to right by 0 to $n-1$. In this new Reflection Puzzle, a digit j, where $j \in \{0,\ldots,(n-1)\}$, can change only when all of the digits from 0 to $j-2$ are zeros, except the leftmost digit which is free to change to any digit at any time.

The possible digit changes, or moves, that are allowed are determined by the following set of rules:

(1) If digit $(j-1) = j$, then j changes to zero.
(2) If j=0, then j changes to equal $j-1$.
(3) If $j \neq (j-1)$ and $j \neq 0$, then j changes $(j-1)+j$ (mod 4) if $(j-1)+j \not\equiv 0$(mod 4) and to 2 if $(j-1)+j \equiv 0$(mod 4).

Alternatively, this set of rules also defines the possible moves in the Reflection Puzzle:

(1) If $(j-1) \equiv j$ (mod 2) or $j = 3$, then j changes to $-(j-1)+j$ (mod 4).
(2) If $(j-1) \not\equiv j$ (mod 2) and $j = 3$, then j changes to $(j-1)+j$ (mod 4).

Example 5.2. *According to the rules of the Reflection Puzzle, the fourth digit from the left in the string 00231 can change from 3 to 1, first digit in the string 1320 can change to 0, 2, or 3, and the third digit in the string 0221 can change from 2 to 0.*

This puzzle can be solved in 2^n-1 moves like the SF Puzzle. Notice that the strings labeling the top edge, the left edge, and the diagonal of the graph of the puzzle are simply the binary reflected Gray code. Consequently, solving the puzzle by going from 00...0 to 11...1, 22...2, or 33...3 takes the same number of moves that it takes to solve Spin-Out, and it is likely that an algorithm that computes the distance between any two vertices in the graph can easily be generated, much like that of the SF Puzzle.

Unfortunately, when I tried to extend this puzzle to correspond to a labeling on K_6^n, I was not able to find a simple modular addition rule to describe the allowable moves.

6. Conclusion

Although I looked at existing puzzles and determined desirable properties for a new puzzle, I was unsuccessful at finding a single idea that extended to all iterated complete graphs of even dimensions. I was also unable to extend Theorem 4.4 to apply to all even iterations of complete graphs of even dimensions. Further research could be done to extend or disprove Theorem 4.4 for all even n, and either define a family of puzzles on all even dimension iterated complete graphs or prove that such a construction is impossible.

REFERENCES

[1] Cull, Paul and E.F. Ecklund, Jr. *Towers of Hanoi and Analysis of Algorithms.* American Mathematical Monthly, Volume 92, Number 6, pp. 407-420. 1985.

[2] Cull, Paul and Ingrid Nelson. *Error-correcting codes on the towers of Hanoi graphs.* Discrete Mathematics 208/209, pp. 157-175. 1999.

[3] Cull, Paul and Ingrid Nelson. *Perfect Codes, NP-Completeness, and Towers of Hanoi Graphs.* Bulletin of the ICA, Volume 26, pp. 13-38. 1999.

[4] Frayer, Christopher. *Perfect One Error Correcting Codes and Complete Iterated Graphs.* Proceedings of the REU Program in Mathematics. NSF and Oregon State University. Corvallis, Oregon. August, 2002.

[5] King, Kathleen. *A New Puzzle Based On The SF Labelling Of Iterated Complete Graphs.* Proceedings of the REU Program in Mathematics. NSF and Oregon State University. Corvallis, Oregon. August, 2004.

[6] Kleven, Stephanie. *Perfect Codes on Odd Dimension Sierpinski Graphs.* Proceedings of the REU Program in Mathematics. NSF and Oregon State University. Corvallis, Oregon. August, 2003.

[7] Pruhs, Kirk. *The SPIN-OUT Puzzle.* ACM SIGCSE Bulletin, Volume 25, Issue 3, pp. 36-38. 1993.

[8] Russell, Pamela. *Perfect One-Error-Correcting Codes On Iterated Complete Graphs: Encoding And Decoding For the SF Labelling.* Proceedings of the REU Program in Mathematics. NSF and Oregon State University. Corvallis, Oregon. August, 2004.

[9] Savage, Carla. *A Survey of Combinatorial Gray Codes.* SIAM Review, Volume 39, Number 4, pp. 605-629. 1997.

SOUTHEAST MISSOURI STATE UNIVERSITY
E-mail address: eaweaver1s@semo.edu