The Snowflake Decoding Algorithm

December 2012

Catherine Walker

Thesis Committee:
Dr. J.B. Nation
Dr. Prasad Santhanam

Graduate Committee:
Dr. George Wilkens
Dr. Monique Chyba
Dr. Rufus Willett
Dr. Mickael Chekroun
Dr. Erik Van Erp

Acknowledgements

I would like to thank my advisor, J.B. Nation, for introducing me to coding theory, making me learn fortran, and welcoming me into his office to talk story. I also thank Prasad Santhanam for his enthusiasm in teaching and suggestions to improve this paper. In addition, I would like to thank each of my professors at the University of Hawaiʻi who have all taught me a great deal. I am also grateful to Manabu Hagiwara and Justin Kong for their ideas and suggestions in the coding seminar. Last but certainly not least I would like to thank my mom for always supporting me in everything I do, even if it means being audience to the same presentation twenty times to help me prepare.

MASTER'S THESIS: THE SNOWFLAKE DECODING ALGORITHM

CATHERINE WALKER

ABSTRACT. This paper describes an automated algorithm for generating a group code using any unitary group, initial vector, and generating set that satisfy a necessary condition. Examples with exceptional complex reflection groups, as well as an analysis of the decoding complexity, are also included.

1. INTRODUCTION

Slepian introduced group codes using orthogonal groups acting on real vector spaces in 1968 [4]. Later, Mittelholzer and Lahtonen [5] published a comprehensive paper on real reflection group coding. Fossorier, Nation, and Peterson [6] then refined these codes by introducing subgroup decoding as an efficient way to decode real reflection group codes. This was extended by Kim, Nation, and Shepler [2] to certain complex reflection groups $\mathbf{G}(r, 1, n)$ acting on $\mathbb{C}^n$. Subgroup decoding does not work for all complex reflection groups, but Kim [1] devised an algorithm to correctly decode these groups.

All of these group codes share the same basic scheme. A finite group $\mathbf{G}$ of isometries acts on a vector space V. The codewords are a subset of the orbit $\mathbf{G}\mathbf{x_0}= \{g\mathbf{x}_0 : g \in \mathbf{G}\}$. A codeword $\mathbf{w} = g^{-1}\mathbf{x}_0$ is transmitted and the received vector is $\mathbf{r} = \mathbf{w} + \mathbf{n}$, where $\mathbf{n}$ represents channel noise. Let $\mathbf{r}_0 = \mathbf{r}$. We recursively apply a transformation c_{k+1} from a specified set $X_{k+1} \subset \mathbf{G}$ to obtain $\mathbf{r}_{k+1} = c_{k+1}\mathbf{r}_k$. The process terminates in a set number m of steps and we decode as $g' = c_m ... c_1$.

In this paper a method to construct effective unitary group codes will be presented. Given any finite unitary group $\mathbf{G}$, a suitable initial vector $\mathbf{x}_0$, and generators X satisfying

$$(\dagger) \text{ for every } \mathbf{w} \in \mathbf{G}\mathbf{x}_0 - \{\mathbf{x}_0\} \text{ there exists } c \in X \text{ such that } \|c\mathbf{w} - \mathbf{x}_0\| < \|\mathbf{w} - \mathbf{x}_0\|$$

it yields a decoding algorithm. For some groups $\mathbf{G}$, with a suitable choice of $\mathbf{x}_0$ and X, decoding algorithms of very low complexity are obtained. Examples using complex reflection groups will also be provided, along with an analysis of the decoding complexity.

2. PRELIMINARIES

All vectors are column vectors and $\mathbf{M}^H$ denotes the conjugate transpose of a complex vector or matrix $\mathbf{M}$.

Definition 2.1. A *unitary matrix* $\mathbf{U}$ is a square complex matrix such that $\mathbf{U}^H\mathbf{U} = I$.

Definition 2.2. A *unitary group* is a group of $n \times n$ unitary matrices, with the group operation of the usual matrix multiplication.

Definition 2.3. The *inner product* is $\langle \mathbf{x}, \mathbf{y} \rangle = \mathbf{x}^H\mathbf{y}$.

Proposition 2.4. *A unitary group acting on a vector space preserves the inner product.*

Date: August 2012.

Proof. Let $\mathbf{U}$ be a unitary matrix and $\mathbf{x}, \mathbf{y}$ be elements of the vector space.

$$\begin{aligned} \langle \mathbf{Ux}, \mathbf{Uy} \rangle &= (\mathbf{Ux})^H \mathbf{Uy} \\ &= \mathbf{x}^H \mathbf{U}^H \mathbf{Uy} \\ &= \mathbf{x}^H \mathbf{y} \\ &= \langle \mathbf{x}, \mathbf{y} \rangle \end{aligned}$$

Thus, the inner product is preserved. □

Definition 2.5. The *norm* of the inner product space is defined as $\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle}$. The distance between vectors $\mathbf{x}$ and $\mathbf{y}$ is given by $\|\mathbf{x} - \mathbf{y}\|$.

Proposition 2.6. *A unitary group acting on a vector space is an isometry with respect to the norm from the inner product.*

Proof. Let $\mathbf{U}$ be a unitary matrix and $\mathbf{x}, \mathbf{y}$ be elements of the vector space.

$$\begin{aligned} \|\mathbf{Ux} - \mathbf{Uy}\| &= \sqrt{\langle \mathbf{Ux} - \mathbf{Uy}, \mathbf{Ux} - \mathbf{Uy} \rangle} \\ &= \sqrt{\langle \mathbf{U}(\mathbf{x} - \mathbf{y}), \mathbf{U}(\mathbf{x} - \mathbf{y}) \rangle} \\ &= \sqrt{\langle \mathbf{x} - \mathbf{y}, \mathbf{x} - \mathbf{y} \rangle} \\ &= \|\mathbf{x} - \mathbf{y}\| \end{aligned}$$

Thus, $\mathbf{U}$ preserves distance and length. □

3. Snowflake Decoding Algorithm

We consider a finite unitary group $\mathbf{G}$ acting on a complex vector space $\mathbb{C}^n$, with a fixed initial vector $\mathbf{x}_0$. As usual, the codewords are a subset of the orbit $\mathbf{Gx_0}= \{g\mathbf{x}_0 : g \in \mathbf{G}\}$. A codeword $\mathbf{w} = g^{-1}\mathbf{x}_0$ is transmitted and the received vector is $\mathbf{r} = \mathbf{w} + \mathbf{n}$, where $\mathbf{n}$ represents noise.

Let $\mathbf{r}_0 = \mathbf{r}$. We recursively apply a transformation c_{k+1} from a specified set $X_{k+1} \subset \mathbf{G}$ to obtain $\mathbf{r}_{k+1} = c_{k+1}\mathbf{r}_k$. After a set number m of steps, terminate and decode as $c_m ... c_1$. The two key parts to the snowflake coding scheme are which elements should be included in each set X_i and how the transformation c_{k+1} should be chosen.

3.1. How to choose c_{k+1}.

The transformation $c_{k+1} \in X_{k+1}$ is chosen such that $\|c_{k+1}\mathbf{r}_k - \mathbf{x}_0\| < d_{k+1} + \epsilon_{k+1}$, where d_i and ϵ_i are predetermined distances that vary for each step. Following is the procedure to determine d_i and ϵ_i.

Let $X \subset \mathbf{G}$ be a generating set for $\mathbf{G}$. In general, this set may include any generating matrices, and powers thereof, elements used in the group's presentation along with their inverses, and the identity. In order for the algorithm to work, X must be chosen so that the condition (†) is satisfied. Let W_0 be the set of all codewords and $d_0 = \max \|\mathbf{w} - \mathbf{x}_0\|$. Define

$$d_1 = \max_{\mathbf{w} \in W_0} \min_{h \in X} \|h\mathbf{w} - \mathbf{x}_0\|.$$

The distance d_1 is the closest all of the codewords can get to the initial vector using any of the provided transformations in X. Now set

$$W_1 = \{\mathbf{w} \in W_0 : \|\mathbf{w} - \mathbf{x}_0\| \leq d_1\}.$$

Thus we get the subset W_1 of codewords which are within the distance d_1 of the initial vector. Since (†) is satisfied, for every codeword $\mathbf{w} \in W_0$ there exists $h \in X$ such that $\|h\mathbf{w} - \mathbf{x}_0\| < \|\mathbf{w} - \mathbf{x}_0\|$. Therefore $d_1 < d_0$ and $W_1 \subset W_0$.

We then recursively get

$$d_{j+1} = \max_{\mathbf{w} \in W_j} \min_{h \in X} \|h\mathbf{w} - \mathbf{x}_0\|.$$

and

$$W_{j+1} = \{\mathbf{w} \in W_j : \|\mathbf{w} - \mathbf{x}_0\| \leq d_{j+1}\}.$$

We continue this process until we get $d_1 > d_2 > \cdots > d_m = 0$, and $W_0 \supset W_1 \supset ... \supset W_m = \{\mathbf{x}_0\}$.

Ideally we would choose c_{k+1} such that $\|c_{k+1}\mathbf{r}_k - \mathbf{x}_0\| < d_{k+1}$. However some tolerance must be built in to account for noise. The tolerance, ϵ_i, should intuitively be half the difference between the distance d_i and distance to $\mathbf{x}_0$ of the nearest codeword outside the ball of radius d_i. For $k > 0$ let

$$e_k = \min_{\substack{\mathbf{w} \in W_0 \\ \|\mathbf{w} - \mathbf{x}_0\| > d_k}} \|\mathbf{w} - \mathbf{x}_0\|.$$

Then $\epsilon_k = \frac{1}{2}(e_k - d_k)$.

3.2. **Determining the set X_{k+1}.**

If we let $X_i = X$ for each step, then the decoding process will terminate; however, it will be much longer than necessary. The program will apply each transformation h in the set then calculate $\|h\mathbf{r}_k - \mathbf{x}_0\|$ to see if it is less than d_{k+1}. The program continues to make these comparisons until a desired transformation is found. Decreasing the number of comparisons by finding ordered sets X_i will increase the efficiency of the decoding process.

Example. Start by applying all transformations in the set X to all codewords in $W_k - W_{k+1} = \{\mathbf{w} \in W_k : \|\mathbf{w} - \mathbf{x}_0\| > d_{k+1}\}$. This will produce an array of codewords and the transformations which will move them to within d_{k+1} of the initial vector. For example, the table below shows a relationship between codewords and transformations. An entry $a_{i,j} = 1$ if $\|h_j\mathbf{w}_i - \mathbf{x}_0\| \leq d_{k+1}$, and zero otherwise.

Table 1. Output for d_{k+1}

	h_1	h_2	h_3	h_4	h_5
$\mathbf{w}_1$	1	0	1	1	1
$\mathbf{w}_2$	1	1	0	0	1
$\mathbf{w}_3$	1	1	1	0	0
$\mathbf{w}_4$	0	0	0	1	0
$\mathbf{w}_5$	0	0	1	0	0
$\mathbf{w}_6$	0	1	0	0	0
$\mathbf{w}_7$	1	0	0	1	0
sum	4	3	3	3	2

Naively, we could choose the transformations based on how many codewords they could be used for. The transformation h_1 is used 4 times, so it would then be the first one chosen. Then if we ignore rows 1, 2, 3, and 7 and sum the remaining rows, we find that h_2, h_3, and h_4 all need to be used once and must be included in the set X_{k+1} as well. The transformation h_5 can be excluded. Let $X'_{k+1} = \{h_1, h_2, h_3, h_4\}$ be an ordered set. If we compute the number of comparisons that need to be made to decode these seven elements in this one round we get one comparison each for codewords $\mathbf{w}_1$,$\mathbf{w}_2$,$\mathbf{w}_3$, and $\mathbf{w}_7$ since the program would choose h_1 for all of them. Then, for codeword $\mathbf{w}_6$, $\mathbf{w}_5$, and $\mathbf{w}_4$ we have 2, 3, and 4 comparisons respectively. This totals to 13 comparisons for this step of decoding.

Now we will look at another way to determine the set X_{k+1}. Instead of considering only the frequency of a transformation, let us also take into account the criticality of the transformation. In Table 1 we can see that h_2, h_3, and h_4 must be used since they are the only transformations available for $\mathbf{w}_6$, $\mathbf{w}_5$, and $\mathbf{w}_4$. Therefore, they are more critical than a transformation which only has overlapping uses. To quantify this idea, consider the weighted table below which has row sums equal to 1. According to Table 2, h_4 should be the first transformation in X_{k+1}. Then, we omit rows 1, 4, and 7 and recompute the sums. We get that column 2 has the highest sum, so h_2 should be the next transformation in X_{k+1}. The only remaining codeword is $\mathbf{w}_5$, thus h_3 is the final transformation in our set. We let $X_{k+1} = \{h_4, h_2, h_3\}$ be the ordered set. Note that $|X_{k+1}| < |X'_{k+1}|$ but more importantly, fewer comparisons are made using the set X rather than the set X'. With X, a total of $(3 \cdot 1) + (3 \cdot 2) + (1 \cdot 3) = 12$ comparisons are made for all codewords in this round.

TABLE 2. Weighted Output for d_{k+1}

	h_1	h_2	h_3	h_4	h_5
$\mathbf{w}_1$	$^1/_4$	0	$^1/_4$	$^1/_4$	$^1/_4$
$\mathbf{w}_2$	$^1/_3$	$^1/_3$	0	0	$^1/_3$
$\mathbf{w}_3$	$^1/_3$	$^1/_3$	$^1/_3$	0	0
$\mathbf{w}_4$	0	0	0	1	0
$\mathbf{w}_5$	0	0	1	0	0
$\mathbf{w}_6$	0	1	0	0	0
$\mathbf{w}_7$	$^1/_2$	0	0	$^1/_2$	0
sum	$^{17}/_{12}$	$^{20}/_{12}$	$^{19}/_{12}$	$^{21}/_{12}$	$^7/_{12}$

The previous example highlighted some of the key concepts for analysis when forming the generating sets X_i. Here is the general algorithm to find the ordered set X_1. Define the function

$$N(\mathbf{w}, d) = |\{g \in X : \|g\mathbf{w} - \mathbf{x}_0\| \leq d\}|$$

which counts the number of transformations which move codeword $\mathbf{w}$ to within d of the initial vector. Then, let

$$B_{1,0}(g) = \{\mathbf{w} \in W_0 - W_1 : \|g\mathbf{w} - \mathbf{x}_0\| \leq d_1\}$$

and

$$S_{1,0}(g) = \sum_{\mathbf{w} \in B_{1,0}(g)} \frac{1}{N(\mathbf{w}, d_1)}$$

Choose h_0 that maximizes $S_{1,0}$, i.e., $S_{1,0}(h_0) \geq S_{1,0}(g)$ for all $g \in X$. Now let

$$B_{1,1}(g) = \{\mathbf{w} \in W_0 - W_1 : \|g\mathbf{w} - \mathbf{x}_0\| \leq d_1\} - B_{1,0}(h_0)$$

$$S_{1,1}(g) = \sum_{\mathbf{w} \in B_{1,1}(g)} \frac{1}{N(\mathbf{w}, d_1)}$$

and choose h_1 that maximizes $S_{1,1}$. Recursively,

$$B_{1,i+1}(g) = \{\mathbf{w} \in W_0 - W_1 : \|g\mathbf{w} - \mathbf{x}_0\| \leq d_1\} - \bigcup_{0 \leq t \leq i} B_{1,t}(h_t)$$

$$S_{1,i+1}(g) = \sum_{\mathbf{w} \in B_{1,i+1}(g)} \frac{1}{N(\mathbf{w}, d_1)}$$

and choose h_{i+1} which maximizes $S_{1,i+1}$. Terminate when $S_{1,r+1} \equiv 0$ or equivalently when $B_{1,r+1}(g) = \emptyset$ for all $g \in X$. The definition of d_1 guarantees that for all codewords, $\mathbf{w}$, there exists $c \in X$ such that $\|c\mathbf{w} - \mathbf{x}_0\| \leq d_1$. Therefore this process will terminate in at most $|X|$ steps. The ordered set $X_1 = \{h_0, h_1, \ldots, h_r\}$ is the ordered set of transformations to be used in the first round.

For X_k we begin again with

$$B_{k,0}(g) = \{\mathbf{w} \in W_{k-1} - W_k : \|g\mathbf{w} - \mathbf{x}_0\| \leq d_k\}$$

and

$$S_{k,0}(g) = \sum_{\mathbf{w} \in B_{k,0}(g)} \frac{1}{N(\mathbf{w}, d_k)}.$$

Choose h_0 that maximizes $S_{k,0}$. Then,

$$B_{k,i+1}(g) = \{\mathbf{w} \in W_{k-1} - W_k : \|g\mathbf{w} - \mathbf{x}_0\| \leq d_k\} - \bigcup_{0 \leq t \leq i} B_{k,t}(h_t)$$

and

$$S_{k,i+1}(g) = \sum_{\mathbf{w} \in B_{k,i+1}(g)} \frac{1}{N(\mathbf{w}, d_k)}$$

Choose h_{i+1} which maximizes $S_{k,i+1}$. Again terminate when $S_{k,r+1} \equiv 0$ to get the set $X_k = \{h_0, h_1, \ldots, h_r\}$

With these sets each codeword now has a decoding canonical form. A codeword $\mathbf{w} = g^{-1}\mathbf{x}_0$ will be decoded as $g = c_m \ldots c_1$ where each c_i is the first element of the ordered set X_i such that $\|c_i \ldots c_1 \mathbf{w} - \mathbf{x}_0\| \leq d_i$.

3.3. **Decoding Theorem.**

Now that the decoding algorithm has been generated we use the following theorem to ensure proper decoding with sufficiently small noise.

Theorem 3.1. *If* $\|\mathbf{r} - \mathbf{w}\| < \eta$, *where* $\mathbf{w} = g^{-1}\mathbf{x}_0$ *and* $\eta = \min(\epsilon_1, \ldots, \epsilon_m)$, *then the Snowflake algorithm decodes* $\mathbf{r}$ *to* g.

Proof. Let $\mathbf{w}$ decode to $g = c_m \ldots c_1$. Let $\mathbf{r}_0 = \mathbf{r}$, $\mathbf{w}_0 = \mathbf{w}$, $\mathbf{r}_{j-1} = c_{j-1} \ldots c_1 \mathbf{r}_0$ and $\mathbf{w}_{j-1} = c_{j-1} \ldots c_1 \mathbf{w}_0$. The received vector $\mathbf{r}$ would not decode to g if at some step c_j is not chosen. This would occur if a preceding transformation $h \neq c_j$ in the ordered set X_j is chosen, or c_j fails to move r_{j-1} to within d_j of $\mathbf{x}_0$.

Suppose $\mathbf{r}$ has decoded correctly through step $j-1$. Note that

$$\|\mathbf{r}_{j-1} - \mathbf{w}_{j-1}\| = \|c_{j-1} \ldots c_1 \mathbf{r}_0 - c_{j-1} \ldots c_1 \mathbf{w}_0\| = \|\mathbf{r}_0 - \mathbf{w}_0\| < \eta.$$

By the triangle inequality,

$$\big|\|h\mathbf{r}_{j-1} - \mathbf{x}_0\| - \|h\mathbf{w}_{j-1} - \mathbf{x}_0\|\big| \leq \|h\mathbf{r}_{j-1} - h\mathbf{w}_{j-1}\| = \|\mathbf{r}_{j-1} - \mathbf{w}_{j-1}\| < \eta \tag{1}$$

for all $h \in \mathbf{G}$.

Consider step j. Suppose some $h \in X_j$ precedes c_j and satisfies $\|h\mathbf{r}_{j-1} - \mathbf{x}_0\| < d_j + \epsilon_j$. However, since h was not chosen for step j by the Snowflake algorithm we know $\|h\mathbf{w}_{j-1} - \mathbf{x}_0\| \geq d_j + 2\epsilon_j$. We then have $\big|\|h\mathbf{w}_{j-1} - \mathbf{x}_0\| - \|h\mathbf{r}_{j-1} - \mathbf{x}_0\|\big| > \epsilon_j \geq \eta$ which contradicts equation (2). Thus, no preceding $h \neq c_j$ can be chosen.

For the other type of error to occur it would mean that $\|c_j\mathbf{r}_{j-1} - \mathbf{x}_0\| \geq d_j + \epsilon_j$. However by definition, $\|c_j\mathbf{w}_{j-1} - \mathbf{x}_0\| \leq d_j$. Therefore, $\big|\|c_j\mathbf{r}_{j-1} - \mathbf{x}_0\| - \|c_j\mathbf{w}_{j-1} - \mathbf{x}_0\|\big| \geq \epsilon_j \geq \eta$ which again contradicts equation (2). Hence, at each step j, c_j is chosen.

By induction, for step m we then have $\|\mathbf{r}_m - \mathbf{w}_m\| = \|c_m \ldots c_1 \mathbf{r} - \mathbf{x}_0\| < \eta \leq \epsilon_m$. Therefore, $\mathbf{r}$ decodes to $c_m \ldots c_1 = g$. Thus for sufficiently small noise the Snowflake algorithm decodes correctly. □

4. EXAMPLES

This section uses irreducible finite complex reflection groups as classified by Shephard and Todd [3]. Using the Snowflake method, the algorithm introduced by Kim [1] to decode exceptional complex reflection groups can be refined. Following some preliminary information, examples of how to create and decode codes using two and three-dimensional complex reflection groups are presented.

Definition 4.1. A matrix $\mathbf{R}$ is a *reflection* if

1. $\mathbf{R}$ is unitary, i.e., $\mathbf{R}\mathbf{R}^H = \mathbf{I}$,
2. $\mathbf{R}$ fixes a hyperplane pointwise.

A matrix $\mathbf{R}$ is a reflection if and only if $\mathbf{R} = \mathbf{I} - (1-\lambda)\mathbf{u}\mathbf{u}^H$ for some λ, $\mathbf{u}$ with $|\lambda| = 1$ and $\mathbf{u}^H\mathbf{u} = 1$.

Definition 4.2. A *complex reflection group* is a unitary group generated by a set of complex reflections which acts on a finite-dimensional complex vector space.

Proposition 4.3. *Conjugating a reflection by any element in a unitary group results in a reflection.*

Proof. Let $\mathbf{G}$ be a reflection group, $\mathbf{R} = \mathbf{I} - (1-\lambda)\mathbf{u}\mathbf{u}^H$ be a reflection in $\mathbf{G}$, and $\mathbf{S}$ be some element in $\mathbf{G}$.

$$\begin{aligned} \mathbf{S}\mathbf{R}\mathbf{S}^H &= \mathbf{S}(\mathbf{I} - (1-\lambda)\mathbf{u}\mathbf{u}^H)\mathbf{S} \\ &= \mathbf{S}\mathbf{S}^H - (1-\lambda)\mathbf{S}\mathbf{u}\mathbf{u}^H\mathbf{S}^H \\ &= \mathbf{I} - (1-\lambda)(\mathbf{S}\mathbf{u})(\mathbf{S}\mathbf{u})^H \end{aligned}$$

which is another reflection. □

Moreover, the inverse and powers of reflections are also reflections fixing the same hyperplane.

4.1. **Example 1:** A two-dimensional group with an easy presentation is $\mathbf{G}_8 : \mathbf{A}^4 = \mathbf{B}^4 = \mathbf{I}$, $\mathbf{ABA} = \mathbf{BAB}$, $|\mathbf{G}_8| = 96$, and it has 18 reflections.

4.1.1. *Constructing the group.* We start by contructing the generators $\mathbf{A}$ and $\mathbf{B}$ which are reflections, so we can write

$$\mathbf{A} = \mathbf{I} - (1-\lambda)\mathbf{u}\mathbf{u}^H$$
$$\mathbf{B} = \mathbf{I} - (1-\lambda)\mathbf{v}\mathbf{v}^H$$

For $\mathbf{G}_8$ we have $\lambda = i$ and can let $\mathbf{u} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}$ so then $\mathbf{A} = \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}$.

To determine $\mathbf{v}$ and construct $\mathbf{B}$ we use the relation $\mathbf{ABA} = \mathbf{BAB}$. Let $c = 1 - \lambda$.

$$\mathbf{ABA} = \mathbf{I} - c\mathbf{u}\mathbf{u}^H - c\mathbf{v}\mathbf{v}^H + c^2(\mathbf{u}^H\mathbf{v})\mathbf{u}\mathbf{v}^H + c^2(\mathbf{v}^H\mathbf{u})\mathbf{v}\mathbf{u}^H + (c^2 - c - c^3|\mathbf{u}^H\mathbf{v}|^2)\mathbf{u}\mathbf{u}^H$$
$$\mathbf{BAB} = \mathbf{I} - c\mathbf{u}\mathbf{u}^H - c\mathbf{v}\mathbf{v}^H + c^2(\mathbf{u}^H\mathbf{v})\mathbf{u}\mathbf{v}^H + c^2(\mathbf{v}^H\mathbf{u})\mathbf{v}\mathbf{u}^H + (c^2 - c - c^3|\mathbf{u}^H\mathbf{v}|^2)\mathbf{v}\mathbf{v}^H$$

To have $\mathbf{ABA} = \mathbf{BAB}$ and $\mathbf{u}\mathbf{u}^H \neq \mathbf{v}\mathbf{v}^H$ we must have $c^2 - c - c^3|\mathbf{u}^H\mathbf{v}|^2 = 0$. Thus,

$$|\mathbf{u}^H\mathbf{v}|^2 = \frac{-\lambda}{(1-\lambda)^2} = \frac{1}{2(1-\cos\theta)}$$

Where $\lambda = e^{i\theta}$. With our choice of $\mathbf{u}$ we get that $|\mathbf{u}^H\mathbf{v}| = v_2$, thus $v_2^2 = \frac{1}{2}$. Using that $\mathbf{v}^H\mathbf{v} = 1$, we then obtain $v_1^2 = \frac{1}{2}$ as well. For $\mathbf{G}_8$ I will therefore be using

$$\mathbf{v} = \begin{bmatrix} \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \quad \text{and} \quad \mathbf{B} = \begin{bmatrix} \frac{1+i}{2} & \frac{1-i}{2} \\ \frac{1-i}{2} & \frac{1+i}{2} \end{bmatrix}.$$

Similar steps can be utilized to determine $\mathbf{v}$ and $\mathbf{B}$ to satisfy other presentations as well. By counting the orbit of a random vector we can ensure that $\mathbf{A}$ and $\mathbf{B}$ generate the entire intended group and not just a homomorphic image.

We use $\mathbf{A}$ and $\mathbf{B}$ to generate our code by taking the orbit of a unit vector, which will be denoted as the *initial vector*, $\mathbf{x}_0$.

4.1.2. *Choosing the Initial Vector.* We would like to choose $\mathbf{x}_0$ to be real and for symmetry we want

$$\begin{aligned}
\|\mathbf{x}_0 - \mathbf{A}\mathbf{x}_0\| &= \|\mathbf{x}_0 - \mathbf{B}\mathbf{x}_0\|,\\
\|(\mathbf{I} - \mathbf{A})\mathbf{x}_0\| &= \|(\mathbf{I} - \mathbf{B})\mathbf{x}_0\|,\\
\|(1-\lambda)\mathbf{u}\mathbf{u}^H\mathbf{x}_0\| &= \|(1-\lambda)\mathbf{v}\mathbf{v}^H\mathbf{x}_0\|,\\
|1-\lambda||\mathbf{u}^H\mathbf{x}_0| &= |1-\lambda||\mathbf{v}^H\mathbf{x}_0|,\\
|\mathbf{u}^H\mathbf{x}_0| &= |\mathbf{v}^H\mathbf{x}_0|. \qquad (2)
\end{aligned}$$

The above will hold true for any reflection group, but let us return to our example $\mathbf{G}_8$. Let $\mathbf{x}_0 = \begin{bmatrix} s \\ t \end{bmatrix}$, then equation (1) gives us

$$\begin{aligned}
\left| [0 \; 1] \begin{bmatrix} s \\ t \end{bmatrix} \right| &= \left| \left[\frac{1}{\sqrt{2}} \; \frac{-1}{\sqrt{2}} \right] \begin{bmatrix} s \\ t \end{bmatrix} \right|,\\
t &= \frac{s-t}{\sqrt{2}},\\
s &= t(1+\sqrt{2}).
\end{aligned}$$

Let $t = 1$ then $\mathbf{x}_0 = \begin{bmatrix} 1+\sqrt{2} \\ 1 \end{bmatrix}$, which would then be normalized.

4.1.3. *The Generating Set X.* Let $X \subset \mathbf{G}$ consist of all the reflections in $\mathbf{G}$, the identity element $\mathbf{I}$, the element in the presentation of $\mathbf{G}$, and its inverse. By proposition 4.3, we can conjugate reflections $\mathbf{A}$ and $\mathbf{B}$ to find the other reflections of $\mathbf{G}$.

The reflections in $\mathbf{G}_8$ are

$$\begin{array}{cccccc}
\mathbf{B} & \mathbf{B}^2 & \mathbf{B}^3 & \mathbf{A} & \mathbf{A}^2 & \mathbf{A}^3\\
\mathbf{BAB}^3 & \mathbf{BA}^2\mathbf{B}^3 & \mathbf{BA}^3\mathbf{B}^3 & \mathbf{B}^2\mathbf{AB}^2 & \mathbf{B}^2\mathbf{A}^2\mathbf{B}^2 & \mathbf{B}^2\mathbf{A}^3\mathbf{B}^2\\
\mathbf{B}^3\mathbf{AB} & \mathbf{B}^3\mathbf{A}^2\mathbf{B} & \mathbf{B}^3\mathbf{A}^3\mathbf{B} & \mathbf{A}^2\mathbf{BA}^2 & \mathbf{A}^2\mathbf{B}^2\mathbf{A}^2 & \mathbf{A}^2\mathbf{B}^3\mathbf{A}^2
\end{array}$$

To satisfy the condition (†) that for each codeword $\mathbf{w} \neq \mathbf{x}_0$ there exists $h \in X$ such that $\|h\mathbf{w} - \mathbf{x}_0\| < \|\mathbf{w} - \mathbf{x}_0\|$, we need more than just the reflections in our generating set X. In addition to the above reflections we include the presentation element $\mathbf{ABA} = \mathbf{BAB}$, and its inverse $\mathbf{A}^3\mathbf{B}^3\mathbf{A}^3$ to form the set of transformations X to be used for decoding.

4.1.4. *The Snowflake Algorithm Results.* Table 3 shows the results of the snowflake algorithm. Not shown in the table is $d_0 = 2.0$ which is the maximum distance of all codewords before the decoding process begins. Following the steps described in section 3.1, we get the distances d_i, and from the procedure in section 3.2 we get the ordered sets X_i.

Table 3. Snowflake Results for $\mathbf{G}_8$

Step i	Distance d_i	Ordered Set X_i
1	1.000	$\{\mathbf{A}^2\mathbf{B}^2\mathbf{A}^2, \mathbf{B}^2\mathbf{A}^2\mathbf{B}^2, \mathbf{BA}^2\mathbf{B}^3, \mathbf{B}^3\mathbf{A}^2\mathbf{B}, \mathbf{A}^3\mathbf{B}^3\mathbf{A}^3, \mathbf{ABA}\}$
2	0.541	$\{\mathbf{A}^3\mathbf{B}^3\mathbf{A}^3, \mathbf{BAB}^3, \mathbf{ABA}^3, \mathbf{A}^3\mathbf{B}^3\mathbf{A}, \mathbf{AB}^3\mathbf{A}^3, \mathbf{ABA}, \mathbf{B}, \mathbf{A}\}$
3	0.000	$\{\mathbf{B}, \mathbf{B}^3, \mathbf{A}, \mathbf{A}^3\}$

Within three rounds all codewords can be decoded.

4.2. **Example 2:** The three-dimensional complex reflection group $\mathbf{G}_{26}$ has the presentation $\mathbf{A}^2 = \mathbf{B}^3 = \mathbf{C}^3 = \mathbf{I}$, $\mathbf{ABAB} = \mathbf{BABA}$, $\mathbf{BCB} = \mathbf{CBC}$, and $\mathbf{AC} = \mathbf{CA}$. The group $\mathbf{G}_{26}$ has 1296 elements, and 33 reflections.

4.2.1. *Constructing the group.* We start with the generators,

$$\mathbf{A} = \mathbf{I} - (1 - (-1))\mathbf{u}\mathbf{u}^H$$

$$\mathbf{B} = \mathbf{I} - (1 - \omega)\mathbf{v}\mathbf{v}^H$$

$$\mathbf{C} = \mathbf{I} - (1 - \omega)\mathbf{t}\mathbf{t}^H$$

Where $\omega = e^{\frac{2\pi i}{3}}$. Following similar steps to the first example we get that to have $\mathbf{ABAB} = \mathbf{BABA}$, with $\mathbf{v}\mathbf{u}^H \neq \mathbf{u}\mathbf{v}^H$ we must have

$$|\mathbf{v}^H\mathbf{u}|^2 = \frac{1-\omega}{2(1-\omega)} = \frac{1}{2}$$

Let $\mathbf{u} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$, then $v_3^2 = \frac{1}{2}$.

To satisfy $\mathbf{AC} = \mathbf{CA}$ with $\mathbf{u} \neq \mathbf{t}$, we must have that $t_3 = 0$. Then to have $\mathbf{BCB} = \mathbf{CBC}$ we find

$$|\mathbf{v}^H\mathbf{t}|^2 = \frac{1}{3}$$

Let $v_1 = 0$, then $v_2^2 = \frac{1}{2}$, $t_2^2 = \frac{2}{3}$, and $t_1^2 = \frac{1}{3}$. Thus,

$$\mathbf{v} = \begin{bmatrix} 0 \\ \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} \quad \text{and} \quad \mathbf{t} = \begin{bmatrix} \frac{1}{\sqrt{3}} \\ -\frac{\sqrt{2}}{\sqrt{3}} \end{bmatrix}.$$

The group generators are hence

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -1 \end{bmatrix}, \mathbf{B} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \frac{1+\omega}{2} & \frac{1-\omega}{2} \\ 0 & \frac{1-\omega}{2} & \frac{1+\omega}{2} \end{bmatrix} \text{ and } \mathbf{C} = \begin{bmatrix} \frac{2+\omega}{3} & \frac{\sqrt{2}(1-\omega)}{3} & 0 \\ \frac{\sqrt{2}(1-\omega)}{3} & \frac{1+2\omega}{3} & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

4.2.2. *Choosing the initial vector.* Again we want $\|\mathbf{A}\mathbf{x}_0 - \mathbf{x}_0\| = \|\mathbf{B}\mathbf{x}_0 - \mathbf{x}_0\| = \|\mathbf{C}\mathbf{x}_0 - \mathbf{x}_0\|$ so, $|\mathbf{u}^H\mathbf{x}_0| = |\mathbf{v}^H\mathbf{x}_0| = |\mathbf{t}^H\mathbf{x}_0|$. If we let $\mathbf{x}_0 = [x\ y\ z]^H$ and use $\mathbf{u}, \mathbf{v}, \mathbf{t}$ found above, then we get

$$y = z(1 + \sqrt{2}) \text{ and } x = z(\sqrt{3} + \sqrt{2} + 2).$$

Let $z = 1$, then $\mathbf{x}_0 = \begin{bmatrix} \sqrt{3} + \sqrt{2} + 2 \\ 1 + \sqrt{2} \\ 1 \end{bmatrix}$, which is yet to be normalized.

4.2.3. *The Snowflake Algorithm Results.* Following the procedures in sections 3.1 and 3.2 produces the distances and sets for decoding $\mathbf{G}_{26}$. Not shown in table 4 is $d_0 = 2$, the maximum distance of all codewords prior to decoding.

Within five rounds all codewords can be decoded.

Table 4. Snowflake Results for $\mathbf{G}_{26}$

Step i	Distance d_i	Ordered Set X_i
1	1.0647	$\{\mathbf{ABC^2BAB^2CB^2A}, \mathbf{BC^2BAB^2CB^2}, \mathbf{CB^2ACBACB^2CB}, \mathbf{CB^2ACB^2ACB^2CB}, \mathbf{BAC^2B^2ACB^2}, \mathbf{B^2ACBAC^2B}, \mathbf{B^2ACB^2AC^2B}, \mathbf{BAC^2BACB^2}, \mathbf{BACBAC^2B^2}, \mathbf{B^2AC^2B^2ACB}, \mathbf{BACB^2AC^2B^2}, \mathbf{B^2AC^2BACB}, \mathbf{CB^2ABC^2}, \mathbf{C^2BAB^2C}\}$
2	0.9466	$\{\mathbf{B^2C^2B^2}, \mathbf{BCB}, \mathbf{BACBAC^2B^2}, \mathbf{B^2AC^2B^2ACB}, \mathbf{ABAB}, \mathbf{B^2AB^2A}, \mathbf{AC}, \mathbf{AB^2C^2BA}, \mathbf{BC^2B^2}, \mathbf{B^2CB}\}$
3	0.7437	$\{\mathbf{ABA}, \mathbf{BACB^2AC^2B^2}, \mathbf{B^2AC^2BACB}, \mathbf{AB^2A}, \mathbf{ABCB^2A}, \mathbf{AB^2C^2BA}, \mathbf{B}, \mathbf{B^2}, \mathbf{BCB}, \mathbf{B^2C^2B^2}, \mathbf{AC}\}$
4	0.4585	$\{\mathbf{B}, \mathbf{B^2}, \mathbf{BCB^2}, \mathbf{B^2C^2B}, \mathbf{BC^2B^2}, \mathbf{ABA}, \mathbf{AB^2A}, \mathbf{B^2CB}, \mathbf{AC}\}$
5	0.0000	$\{\mathbf{A}, \mathbf{B}, \mathbf{B^2}, \mathbf{C}, \mathbf{AC}, \mathbf{C^2}, \mathbf{AC^2}\}$

5. Analysis of decoding complexity.

One way to measure the efficiency of a decoding algorithm is the average number of comparisons, γ for decoding, then calculate the comparisons per bit, $\delta = \frac{\gamma}{\log_2 |\mathbf{G}|}$. For permutation codes on S_n, the theoretical lower bound for δ is 1 [11]. By utilizing the group structure, the Snowflake algorithm is able to produce group codes with δ less than 1. To get an idea of how the Snowflake algorithm affects δ we will compare values for some complex reflection group codes using several schemes. In addition, δ values for several similar groups using various decoding methods are presented.

5.1. **Improvement in complex reflection group codes.** Table 5 lists the dimension and order for the group, along with the calculations γ and δ using three different decoding schemes. Scheme 1 uses the entire set X for each step, choosing c_{k+1} which minimizes $\|c_{k+1}r_k - \mathbf{x}_0\|$. Scheme 2 uses the entire set X for each step, but uses the distances as described in section 3.1 to choose $c_{k+1} \in X$ such that $\|c_{k+1}r_k - \mathbf{x}_0\| \leq d_{k+1}$. Scheme 3 uses the snowflake algorithm in its entirety as detailed in sections 3.1 and 3.2.

Table 5

			Scheme 1		Scheme 2		Snowflake	
Group	Dim.	Order	γ	δ	γ	δ	γ	δ
$\mathbf{G}_4$	2	24	9.38	2.04	5.83	1.27	4.21	0.92
$\mathbf{G}_5$	2	72	20.78	3.37	12.08	1.96	5.76	0.93
$\mathbf{G}_8$	2	96	24.66	3.74	11.13	1.69	6.19	0.94
$\mathbf{G}_{16}$	2	600	95.14	10.31	55.07	6.0	16.8	1.82
$\mathbf{G}_{20}$	2	360	63.62	7.49	47.0	5.53	8.86	1.04
$\mathbf{G}_{26}$	3	1296	77.08	7.45	47.2	4.56	15.3	1.48

As the table shows, the number of comparisons per bit is greatly reduced using the snowflake algorithm. From scheme 1 to the Snowflake algorithm there is a 55 to 82% decrease in δ, depending upon the group. Even between scheme 2, which uses a portion of the snowflake scheme, compared to the entire Snowflake algorithm we get a reduction in δ of 28 to 81%. Through manipulation of the algorithm choices of the sets X_i by hand, it is possible to make slight improvements for some of the groups. For example, for $\mathbf{G}_4$ we can get $\delta = 0.81$, and for $\mathbf{G}_8$, $\delta = 0.85$. However, this is a tedious process even with small groups for only a minor decrease in δ as compared to the vast improvement from applying the Snowflake algorithm.

5.2. **Comparison of similar groups.** Table 6 lists a group $\mathbf{G}$, the decoding method used, the order of $\mathbf{G}$, and the value δ. The modified insertion sort is described by Fossorier *et al.* [6] and produces a lower δ value than the usual insertion sort.

TABLE 6

Group	Decoding method	$\lvert\mathbf{G}\rvert$	δ
S_8	insertion sort	40,320	1.26
S_8	modified insertion	40,320	1.04
S_{16}	insertion sort	16!	1.64
S_{16}	modified insertion	16!	1.17
S_{32}	insertion sort	32!	2.35
S_{32}	modified insertion	32!	1.47
$\mathbf{G}(4,1,4)$	modified insertion	$4^4 \cdot 4!$	0.69
$\mathbf{G}(4,1,16)$	modified insertion	$4^{16} \cdot 16!$	0.89
$\mathbf{G}(8,1,4)$	modified insertion	$8^4 \cdot 4!$	0.52
$\mathbf{G}(8,1,16)$	modified insertion	$8^{16} \cdot 16!$	0.73
$\mathbf{G}_4$	Snowflake	24	0.92
$\mathbf{G}_5$	Snowflake	72	0.93
$\mathbf{G}_8$	Snowflake	96	0.94
$\mathbf{G}_{16}$	Snowflake	600	1.82
$\mathbf{G}_{20}$	Snowflake	360	1.04
$\mathbf{G}_{26}$	Snowflake	1296	1.48
$\mathbf{G}_8^8 \wr S_8$	Snowflake and mod. insert.	$96^8 \cdot 8!$	0.96
$\mathbf{G}_{20}^{20} \wr S_{16}$	Snowflake and mod. insert.	$360^{16} \cdot 16!$	1.07

Nation and Walker [7] formulated a group code using a group $\mathbf{G}$ of unitary matrices to form the wreath product $\mathbf{G}^n \wr \mathbf{S}_n$. The size of $\mathbf{G}^n \wr \mathbf{S}_n$ is $|\mathbf{G}|^n n!$, which is significantly larger than either group code, $\mathbf{G}$ or $\mathbf{S}_n$, individually, though when combined, the new code has only a slight increase in δ. The last two groups in table 6 are examples of such wreath products, and are decoded using a combination of the Snowflake algorithm and the modified insertion sort. As you can see in the table, several of the group codes using the Snowflake algorithm, including $\mathbf{G}_8^8 \wr \mathbf{S}_8$, achieve $\delta < 1$, whereas the insertion sort must have $\delta \geq 1$. This makes the complex reflection group codes a competetive alternative to permutation codes which use an insertion sort.

The Snowflake algorithm greatly decreases the decoding complexity for complex reflection groups, however they still do not exceed other known codes. A group code using the wreath product of cyclic groups $G(r, 1, n)$, has approximately the same number of codewords and similar minimum distance but runs noticeably faster than the wreath code mentioned above. There are methods used to optomize the codes for $G(r, 1, n)$, which could be applied to $\mathbf{G}^n \wr \mathbf{S}_n$ to further improve these codes as well.

For some reflection groups there is an alternative decoding method outlined by Hagiwara and Wadayama [8] and further discussed by Hagiwara and Kong [9] using LP-decoding of permutation codes. These codes do not produce a canonical form like the Snowflake algorithm does, but they are otherwise quite efficient. A δ value has not yet been calculated for the LP-decoding of permutation codes.

Though it is often a prominent topic in code discussions, speed and low complexity may not be the only desired attribute of a coding scheme. For example, in cryptography other code properties may be more desirable than speed. Memory storage may require more stability than efficiency as well, so different codes may be utlized. For example, Jiang *et al.* [10] describe an algorithm utilizing permutations for use with flash memory cells. While their algorithm has benefits such as a canonical form, the use of an insertion sort leaves room for improvement in efficiency.

6. Future Research

This coding scheme may be made more efficient and robust by utlizing a subset of the group rather than the entire group to make the code. Particular codewords which require more steps and/or comparisons to decode could be omitted entirely to decrease the decoding complexity and possibly allow for a higher noise tolerance. Variations in the generating set X may also improve the code and lead to lower δ values.

The Snowflake decoding algorithm can be used for any unitary group, thus further research could also include finding specific codes for other groups. Some unitary groups, like the Mathieu groups, have thus far been unused for coding. With this algorithm they may be used to make efficient group codes. Other group codes could also be concatenated to form larger codes with little change in decoding complexity.

The modified insertion sort lowers the decoding complexity for permutation codes. Applying the Snowflake algorithm to S_n may produce a further improved insertion sort with an even lower δ value. The algorithm described in this paper is so versatile that it could potentially be used in a variety of capacities, all of which should be explored.

References

[1] Hye Jung Kim, *Complex Reflection Group Coding*, Master's Thesis, Univ. of Hawaii (2011)

[2] H.J. Kim, J.B. Nation, and A. Shepler *Group Codes with Complex Reflection Groups*, Univ. of Hawaii and Univ. of N. Texas

[3] G.C. Shephard and J.A. Todd, *Finite unitary reflection groups*, Canad. J. Math., (1954), 274-304.

[4] D. Slepian, *Group codes for the Gaussian channel*, Bell Syst. Tech. J. 47 (1968), 575-602

[5] T. Mittelholzer and J. Lahtonen, *Group codes generated by finite reflection groups*, IEEE Trans. on Information Theory 42 (1996), 519-528

[6] M. Fossorier, J.B. Nation, and W. Peterson, *Reflection group codes and their decoding*, IEEE Trans. on Information Theory 56 (2010), 6273-6293

[7] J.B. Nation and C. Walker, *Group codes based on wreath products of complex reflection groups* (preprint)

[8] M. Hagiwara and T. Wadayama, *LP decodable permutation codes based on linearly constrained permutation matrices*, Proceedings ISIT (2011), 139-143

[9] M. Hagiwara and J. Kong, *Kendall tau linear programming decodable permutation codes* (preprint)

[10] A.Jiang, R. Mateescu, M. Schwartz, and J. Bruck, *Rank modulation for flash memory*, Proceedings IEEE ISIT (2008), 1731-1735

[11] D. Knuth, *Searching and Sorting, the art of computer programming*, vol. 3, Reading, MA, Addison-Wesley, 1973

University of Hawai'i
E-mail address: `walkercl@hawaii.edu`