2015

Minimizing the Cost of Guessing Games

Lindsay Czap
Grand Valley State University, czapl@mail.gvsu.edu

Minimizing the cost of guessing games

Lindsay Czap
Grand Valley State University

Abstract

A two-player "guessing game" is a game in which the first participant, the "Responder," picks a number from a certain range. Then, the second participant, the "Questioner," asks only yes-or-no questions in order to guess the number.

We will introduce guessing games in which the Responder is allowed to lie. Guessing games with lies are closely linked to error correcting codes, which are mathematical objects that allow us to detect an error in the information that we receive and correct these errors. We will give basic definitions in coding theory and show how error correcting codes allow us to still guess the correct number even if one lie is involved.

We will then introduce cost functions to guessing games. By assigning a "cost" to the games, instead of minimizing the number of questions asked, we must find a way to minimize the total cost of our game. We will discuss optimization methods for minimizing the total cost of a guessing game when the cost function is applied to the answers or when the cost function is applied to the questions.

1 Introduction

The game "20 Questions" has always been a staple for children trapped in long car rides. This game involves two players: the first player picks an object and the second player asks "yes or no" questions in order to guess which object the first picked. The game alternates between questions and answers so questions may be constructed based off of previous answers. This type of adaptive guessing game is called an *online* game. For example, you would not ask the question "Does it have four legs?" if you have already received a "yes" to the question "Is your object a vegetable?"

"20 Questions" requires both players to be present and for each response to be given immediately after its corresponding question. But what if you cannot receive responses immediately after each question? In order to receive all of the responses that you need to guess to correct object, you must ask all of the questions at once and then receive all of the responses at once. This means that we must construct *all* of the questions ahead of time and we cannot adapt questions based upon previous responses. These

types of non-adaptive guessing games are called *offline* games and are the focus of this paper.

In our offline guessing games, the two players decide on a range of integers starting at 0. Then the first player, the "Responder," secretly chooses one of those integers. The second player, the "Questioner," then constructs all of the questions that they wish to ask the Responder. They ask all of these questions at once.

The following is an example of one such guessing game:

Questioner:

Question 1: "Is your number in $\{4, 5, 6, 7\}$?"

Question 2: "Is your number in $\{2, 3, 6, 7\}$?"

Question 3: "Is your number in $\{1, 3, 5, 7\}$?"

Responder:

Response 1: "Yes"

Response 2: "Yes"

Response 3: "No"

The Questioner can then use the Responder's responses to figure out which integer was chosen. A response of "yes" to Question 1 eliminates the possibilities $\{0, 1, 2, 3\}$. A response of "yes" to Question 2 eliminates the possibilities $\{4, 5\}$. A response of "no" to Question 3 eliminates the possibility $\{7\}$. Thus, the Questioner knows that the Responder originally chose 6 as their integer.

2 Background

The guessing games that we will be analyzing have a special property: The Responder is allowed to *lie* to the Questioner. However, even though the Responder is allowed to lie, these games are designed so that the Questioner can *still* guess the correct number that the Responder has chosen.

In order to begin studying these special games, we must first introduce some coding theory definitions. A binary Error Correcting Code (ECC) is a set of binary vectors that has the capacity to detect and then correct for errors that can occur when a message is sent over a noisy channel. These factors can include noise over a digital channel, a misprint in written text, or-in the case of our guessing game-a player lying.

The metric that is defined to measure the distance between two codewords in a code is called the *Hamming Distance*. The Hamming Distance between vectors v_1, v_2, is the number of places in which v_1 and v_2 differ, denoted by $d(v_1, v_2)$. For example, given the codewords v_1 and v_2 with binary vectors 1101 and 1011, respectively, they would differ in 2 places as shown in Figure 1.

v_1	1	1	0	1
v_2	1	0	1	1

Figure 1: Codewords v_1 and v_2.

and thus, $d(v_1, v_2) = 2$. The *Minimum Distance* of a code C is the minimum Hamming Distance between any two codewords in a code, denoted $d(C)$ [3]. The capacity for an ECC to detect and correct for errors is the minimum distance of that code. An ECC C can detect and correct for up to ℓ errors if $d(C) \geq 2\ell + 1$. For more definitions and results on Error Correcting Codes see [3].

$C =$

v_1	1	0	0	1	1	0
v_2	0	1	1	1	0	1
v_3	1	0	1	0	0	1
v_4	1	1	0	0	0	0

Figure 2: The code C.

For example, given the code C in Figure 2, it follows that $d(C) = 3$ since the Hamming Distance between any two codewords in C is at least 3. Thus, C can detect up to $\ell = 1$ lies.

In a guessing game, we will represent the answers to each of our questions using binary vectors. We will assign an answer of "no" to the digit 0 and an answer of "yes" to the digit 1.

Example: In a guessing game, if we asked 4 consecutive questions and their answers were "no", "yes","yes","no", we would represent these responses with the binary vector: 0110.

Since each of the m possible integers of a guessing game are unique, if carefully chosen questions are asked in order to guess the Responder's secretly chosen integer, then the vector composed of the

responses to each of the questions asked would be unique for each possible integer. Thus, each of the m possible integers would be assigned to a unique binary vector that are created based upon the questions that we choose to ask.

Definition 1. *An (m, c, ℓ) guessing game is an offline guessing game with m possible integers and cost function c, that is able to correct for up to ℓ errors.*

Given an (m, c, ℓ) guessing game, let $\mathcal{V}$ be the set of m unique binary vectors that we choose to each represent one of the m possible integers of G and let Q be the set of n questions that are asked about the integers such that $d(\mathcal{V}) \geq 2\ell + 1$.

A guessing game G with cost function $c(\mathcal{V})$ is an offline guessing game in which each of the possible answers are assigned a "price." To ask a question about a different possible answer, the Questioner will be "charged" the cost of all of the possible answers that they are asking about. For example, the cost of the question "Is your number in $\{1, 3\}$?" will be the sum of the cost of asking about 1 and asking about 3. We will let k_G be the sum of the costs of all questions in a guessing game and therefore total cost of that game.

Definition 2. *Given an (m, c, ℓ) guessing game G, $[G]$ is an $m \times n$ guessing game matrix in which each of the m rows of $[G]$ is a vector in $\mathcal{V}$.*

By choosing the vectors in $\mathcal{V}$, we create the guessing game matrix $[G]$ that represents our game G. We can choose $\mathcal{V}$ to be any set of m unique binary vectors that satisfy $d(\mathcal{V}) \geq 2\ell + 1$. Thus, for example, Figure 3 shows one possible matrix for a $(4, c, 1)$ game. We created this matrix by choosing $\mathcal{V}_a = \{00000, 01110, 10101, 11011\}$.

$[G_a] =$

	q_1	q_2	q_3	q_4	q_5
v_1	0	0	0	0	0
v_2	0	1	1	1	0
v_3	1	0	1	0	1
v_4	1	1	0	1	1

Figure 3: A possible game matrix, $[G]_a$, for a $(4, c, 1)$ game.

Similarly, Figure 4 shows another possible matrix for a $(4, c, 1)$ game. We created this second matrix by choosing $\mathcal{V}_b = \{1110101, 0101011, 1011110, 0101100\}$.

$G_b =$

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
v_1	1	1	1	0	1	0	1
v_2	0	1	0	1	0	1	1
v_3	1	0	1	1	1	1	0
v_4	0	1	0	1	1	0	0

Figure 4: Another possible game matrix, $[G]_b$, for a $(4, c, 1)$. game.

Both of the guessing game matrices $[G]_a$ and $[G]_b$ are created by choosing valid sets of representative binary vectors and therefore are both valid matrices representing a $(4, c, 1)$ game.

Given an (m, c, ℓ) guessing game G, choosing any valid set $\mathcal{V}_k$ as our set of binary vectors will create the game matrix $[G]_k$. Each question vector $q_i \in Q$ is determined by the ith column of the matrix $[G]_k$.

For example, we saw that choosing $\mathcal{V}_a$ as our set of binary vectors creates the guessing game matrix $[G]_a$. As shown in Figure 5, we have highlighted the 4th column of $[G]_a$.

	q_1	q_2	q_3	q_4	q_5
v_1	0	0	0	0	0
v_2	0	1	1	1	0
v_3	1	0	1	0	1
v_4	1	1	0	1	1

Figure 5: Column 4 of $[G]_a$.

Thus, we say that this highlighted column is the question vector q_4. Each question in a guessing game asks about all of the codewords whose binary vectors have a 1 in the column of $[G]$ that corresponds to that question. Thus, we see that q_4 is the column vector in Figure 6.

0
1
0
1

Figure 6: The column q_4 of $[G]$.

Therefore, this would be the 4th question in the game G and would be asked as: "Is your number in $\{v_2, v_4\}$? because these are the codewords whose representative binary vector had a 1 in column 4 of $[G]_a$.

3 Results

3.1 Costs on Answers with an Unrestricted Number of Questions

The main focus of this paper is to expand upon the topic of guessing games with lies by adding cost functions.

Definition 3. *Given an (m, c, ℓ) guessing game G with $\mathcal{V} = \{v_1, v_2, ..., v_m\}$, we write the function c as $c(\mathcal{V}) = (c(v_1), c(v_2), ..., c(v_m))$ and, without loss of generality, we assume $c(v_1) \leq c(v_2) \leq ... \leq c(v_m)$.*

Remark. Given a an (m, c, ℓ) guessing game G, the total cost of a game k_G is calculated by $k_G = |v_1|c(v_1) + |v_2|c(v_2) + ... + |v_m|c(v_m)$.

Example. Figure 7 shows a game matrix $[G]$ that defines a $(4, c, 1)$ game with $c(\mathcal{V}) = (2, 4, 6, 8)$.

$[G] =$

	q_1	q_2	q_3	q_4	q_5	q_6
v_1	1	0	0	1	1	0
v_2	0	1	1	1	0	0
v_3	1	0	1	0	0	0
v_4	1	1	0	0	0	0

Figure 7: Game matrix $[G]$.

then the total cost of this game would be calculated by

$$\begin{aligned} k_G &= 3c(v_1) + 3c(v_2) + 2c(v_3) + 2c(v_4) \\ &= 3 \cdot 2 + 3 \cdot 4 + 3 \cdot 6 + 2 \cdot 8 \\ &= 46. \end{aligned}$$

For any (m, c, ℓ) guessing game G, either $\vec{0} \in \mathcal{V}$ or $\vec{0} \notin \mathcal{V}$. In the following portion of the paper, we will show two strategies that we can use to minimize the total cost of a guessing game, depending upon the inclusion or exclusion of $\vec{0}$.

Definition 4. *An $m \times n$ $(3, 0)$ matrix $[G]$ has the following form:*

v_1	*1*	*1*	*1*	*0*	*0*	*0*	*0*	*...*	*0*
v_2	*0*	*0*	*1*	*1*	*1*	*0*	*0*	*...*	*0*
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_{m-1}	*0*	*0*	*0*	*0*	*...*	*0*	*1*	*1*	*1*
v_m	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*	*0*

Thus, if we have answer vectors $v_i, v_j \in \mathcal{V}$ where $|v_i| \geq 3$ and $|v_j| \geq 3$, we may arrange them as follows to assure that $d(v_i, v_j) \geq 3$:

v_i	...	1	1	1	0	0	0	0...
v_j	...	0	0	1	1	1	0	0...

Thus, $d(v_i, \vec{0}) \geq 3$ for every $v_i \in \mathcal{V}$ such that $v_i \neq \vec{0}$ will guarantee $d(\mathcal{V}) \geq 3$.

Definition 5. *An $m \times n$ $(2,1)$ matrix $[G]$ has the following form:*

v_1	1	1	0	0	0	0	0	...	0
v_2	0	0	1	1	0	0	0	...	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_{m-1}	0	0	0	0	...	0	1	1	0
v_m	0	0	0	0	0	0	0	0	1

Given a guessing game G with a $(2,1)$ construction, in order to decrease the total cost k_G of G, we would have to decrease the weight of the answer vectors. Decreasing the weight of any vector v_i from $|v_i| = 2$ to $|v_i| < 2$ would decrease $d(v_i, \vec{0}) = 3$ to $d(v_i, \vec{0}) < 3$ which would not allow for $\ell = 1$.

Remark. For both the $(3,0)$ and $(2,1)$ matrices, it is the case that $d(G) \geq 3$.

Lemma 1. *For any $(m, c, 1)$ game, with the requirement that $\vec{0} \in \mathcal{V}$ then a $(3,0)$ matrix will produce the cheapest guessing game with total cost calculated by $k_{\vec{0}\in\mathcal{V}} = 3 \cdot c(v_1) + 3 \cdot c(v_2) + ... + 3 \cdot c(v_{m-1})$.*

Proof. Let G be an offline $(m, c, 1)$ game with $c = c(v_1), c(v_2), ..., c(v_m))$ and the requirement that $\vec{0} \in \mathcal{V}$ and. Let $[G]$ be a $(3,0)$ matrix and let $k_{\vec{0}\in\mathcal{V}}$ be the minimum cost among all choices of $\mathcal{V}$ such that $\vec{0} \in \mathcal{V}$.

Since $[G]$ is a $(3,0)$ matrix, it follows that the total cost of G, k_G, is calculated by

$$k_G = 3 \cdot c(v_1) + 3 \cdot c(v_2) + ... + 3 \cdot c(v_{m-1}).$$

Since k_G is the total cost of G and $k_{\vec{0}\in\mathcal{V}}$ is the minimum possible cost of any game with $\vec{0} \in \mathcal{V}$, it follows that

$$k_{\vec{0}\in\mathcal{V}} \leq k_G.$$

Since $c(v_m)$ is the maximum cost value in $c(\mathcal{V})$, we claim that any $\mathcal{V}$ giving the minimum cost must have $v_m = \vec{0}$. Since $\ell = 1$, it must be the case that $d(\mathcal{V}) \geq 2\ell + 1 = 3$. Since $\vec{0} \in \mathcal{V}$ it follows that $|v_i| \geq 3$ for all $\vec{0} \neq v_i \in \mathcal{V}$ in order to maintain the minimum distance of the guessing game code. Choosing $|v_i| = 3$ for every $v_i \neq \vec{0}$ will minimize k_G. Thus we can see that k_G is minimized when $|v_m| = 0$ and $|v_i| = 3$ for every $v_i \in \mathcal{V}$ where $v_i \neq \vec{0}$.

Suppose we let $v_i = \vec{0}$, k_i be the cost when $v_i = \vec{0}$, and k_m be the cost when $v_m = \vec{0}$. It follows that $k_i = c(v_1) \cdot |v_1| + c(v_2) \cdot |v_2| + ... + c(v_{i-1}) \cdot |v_{i-1}| + c(v_i) \cdot |0| + c(v_{i+1}) \cdot |v_{i+1}| + ... + c(v_m) \cdot |v_m|$ and $k_m = c(v_1) \cdot |v_1| + c(v_2) \cdot |v_2| + ... + c(v_{m-1}) \cdot |v_{m-1}| + c(v_{m-1}) \cdot |0|$. Therefore,

$$k_i - k_m = c(v_m)|v_m| - c(v_i)|v_i|.$$

Since we have shown that for every $v_j \neq \vec{0} \in \mathcal{V}$, $|v_j| = 3$, and $c(v_1) \leq c(v_2) \leq ... \leq c(v_m)$, then it follows that $k_i - k_m \geq 0$ and therefore $k_i \geq k_m$. Therefore, choosing $v_m = \vec{0}$ will either decrease or not change the cost and thus we have proven our claim and we will choose $v_m = \vec{0}$. Thus, when $\vec{0} \in \mathcal{V}$, the minimum cost satisfies $k_{\vec{0} \in \mathcal{V}} \geq k_G$.

Since we have shown that $k_{\vec{0} \in \mathcal{V}} \leq k_G$ and $k_{\vec{0} \in \mathcal{V}} \geq k_G$, it follows that

$$k_{\vec{0} \in \mathcal{V}} = k_G = 3 \cdot c(v_1) + 3 \cdot c(v_2) + ... + 3 \cdot c(v_{m-1})$$

and this is realized by a $(3, 0)$ matrix. □

Lemma 2. *For any $(m, c, 1)$ game, if $\vec{0} \notin \mathcal{V}$ then a $(2, 1)$ construction will produce the cheapest guessing game with total cost calculated by $k_{\vec{0} \notin \mathcal{V}} = 2 \cdot c(v_1) + 2 \cdot c(v_2) + ... + 2 \cdot c(v_{m-1}) + 1 \cdot c(v_m)$.*

Proof. Let G be an offline $(m, c, 1)$ game with $c = (cv_1), c(v_2), ..., c(v_m))$ and the requirement that $\vec{0} \notin \mathcal{V}$ and. Let $[G]$ be a $(2, 1)$ matrix and let $k_{\vec{0} \notin \mathcal{V}}$ be the minimum cost among all choices of $\mathcal{V}$ such that $\vec{0} \notin \mathcal{V}$.

Since $[G]$ is a $(2, 1)$ matrix, it follows that the total cost of G, $k_{(2,1)}$, is calculated by

$$k_{(2,1)} = 2 \cdot c(v_1) + 2 \cdot c(v_2) + ... + 2 \cdot c(v_{m-1}) + 1 \cdot c(v_m).$$

Since $k_{(2,1)}$ is the total cost of G and $k_{\vec{0}\notin\mathcal{V}}$ is the minimum possible cost of any game with $\vec{0} \in \mathcal{V}$, it follows that

$$k_{\vec{0}\notin\mathcal{V}} \leq k_{(2,1)}.$$

Since $\vec{0} \notin \mathcal{V}$, we choose $v_j \in \mathcal{V}$ such that $|v_j| = 1$. Suppose such a vector v_j does not exist. Therefore, it would follow that $|v_i| \geq 2$ for every $v_i \in \mathcal{V}$.

However, we can replace v_i with a vector of weight 1 and still maintain $d(\mathcal{V}) \geq 3$. We assume that the set of vectors $v_1, ..., v_i, ..., v_m$ in Figure 3.1 make $d(\mathcal{V}) \geq 3$. By changing v_i to $\vec{0}$ and asking one more question about only v_i in Figure 3.1 guarantees that $|v_i| \geq 2$ can be changed to $|v_i| = 1$ while maintaining $d(\mathcal{V}) \geq 3$.

-	-	-	-	-	v_1	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
-	-	-	-	-	v_i	-	-	-	-
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
-	-	-	-	-	v_m	-	-	-	-

[	-	-	-	-	-	v_1	-	-	-	-	]	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
[	-	-	-	-	-	v_i	-	-	-	-	]	1
⋮		⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
[	-	-	-	-	-	v_m	-	-	-	-	]	0

Since we have decreased the weight of one answer vector while every other component of the game remains constant, we have therefore decreased our cost. Therefore, in order to minimize c, there we will choose a $v_j \in \mathcal{V}$ such that$|v_j| = 1$.

Next we claim that there cannot exist two vectors $v_a, v_b \in \mathcal{V}$ such that $|v_a| = |v_b| = 1$.

Suppose such vectors exist. It would then follow that $d(v_a, v_b) \leq 2$, and thus $d(\mathcal{V}) \leq 2$. Therefore, only one answer vector of weight 1 can be included in $\mathcal{V}$.

We claim that we should assign our answer vectors such that $|v_m| = 1$ since $c(v_m) \geq c(v_i)$ for all $v_i \in \mathcal{V}$.

Consider the total cost with $|v_i| = 1$ for some $1 \leq i \leq m$:

$$k = c(v_1)|v_1| + c(v_2)|v_2| + ...c(v_i) \cdot 1 + ... + c(v_m)|v_m|.$$

Also consider the total cost with $|v_m| = 1$

$$k' = c(v_1)|v_1| + c(v_2)|v_2| + ...c(v_i)|v_i| + ... + c(v_m) \cdot 1.$$

We can see that

$$\begin{aligned} k - k' &= c(v_i)(1 - |v_m|) + c(v_m)(|v_m| - 1) \\ &= (|v_m| - 1)(c(v_m) - c(v_i)). \end{aligned}$$

Since $|v_m| \geq 2$, it follows that $|v_m| - 1 > 0$. Since $c(v_m)$ is the largest cost value in G, it follows that $c(v_m) - c(v_i) \geq 0$.

Thus, it follows that $k - k' \geq 0$ and $k \geq k'$ and therefore this means that assigning a weight of 1 to the vector v_m will either decrease the total cost or not change the total cost. Thus, k_G is minimized when $|v_m| = 1$ and $|v_i| = 2$ for every $v_i \neq v_m$.

Thus, when $\vec{0} \notin \mathcal{V}$, the minimum cost satisfies $k_{\vec{0} \notin \mathcal{V}} \geq k_{(2,1)}$.

Since we have shown that $k_{\vec{0} \notin \mathcal{V}} \leq k_{(2,1)}$ and $k_{\vec{0} \notin \mathcal{V}} \geq k_{(2,1)}$, it follows that

$$k_{\vec{0} \notin \mathcal{V}} = k_{(2,1)} = 2 \cdot c(v_1) + 2 \cdot c(v_2) + ... + 2 \cdot c(v_{m-1}) + 1 \cdot |v_m|$$

and this is realized by a $(2, 1)$ matrix. □

Theorem 3. *If $c(v_m) < \sum_{i=1}^{m-1} c(v_i)$ then the minimum cost of an $(m, c, 1)$ game G is realized by a $(3, 0)$ matrix. And if $c(v_m) > \sum_{i=1}^{m-1} c(v_i)$ then the minimum cost is realized by a $(2, 1)$ matrix for any cost function.*

Proof. Let G be a $(m, c, 1)$ offline guessing game with $c(\mathcal{V}) = (c(v_1), c(v_2), ..., c(v_m))$. Consider the case where $c(v_m) < \sum_{i=1}^{m-1} c(v_i)$.

By Lemmas 1 and 2, it follows that

$$
\begin{aligned}
k_{\vec{0}\notin\mathcal{V}} &= c(v_1)\cdot 2 + c(v_2)\cdot 2 + ... + c(v_m)\cdot 1 \\
&< c(v_1)\cdot 2 + c(v_2)\cdot 2 + ... + \sum_{i=1}^{m-1} c(v_i)\cdot 1 \\
&= c(v_1)\cdot c(v_1)\cdot 2 + c(v_2)\cdot c(v_2)\cdot 2 + ... + c(v_{m-1})\cdot c(v_{m-1})\cdot 2 \\
&= c(v_1)\cdot 3 + c(v_2)\cdot 3 + ... + c(v_{m-1})\cdot 3 \\
&= k_{\vec{0}\in\mathcal{V}}
\end{aligned}
$$

Therefore, $c(v_m) < \sum_{i=1}^{m-1} c(v_i)$ implies $k_{\vec{0}\notin\mathcal{V}} < k_{\vec{0}\in\mathcal{V}}$. Similarly, $c(v_m) > \sum_{i=1}^{m-1} c(v_i)$ implies $k_{\vec{0}\notin\mathcal{V}} > k_{\vec{0}\in\mathcal{V}}$. □

Example.

Let $G_1 = (4, c_1, 1)$ and $G_2 = (4, c_2, 1)$ where $c_1 = (1, 2, 3, 4)$ and $c_2 = (1, 2, 3, 10)$.

The two possible best strategies for G_1 would be:

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
v_1	1	1	1	0	0	0	0
v_2	0	0	1	1	1	0	0
v_3	0	0	0	0	1	1	1
v_4	0	0	0	0	0	0	0

$k_{G_1} = 3(1) + 3(2) + 3(3) = 18$

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
v_1	1	1	0	0	0	0	0
v_2	0	0	1	1	0	0	0
v_3	0	0	0	0	1	1	0
v_4	0	0	0	0	0	0	1

$k_{G_1} = 2(1) + 2(2) + 2(3) + 1(4) = 16$

In the case of G_1, the $(2, 1)$ matrix realizes the cheapest game.

The two possible best strategies for G_2 would be:

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
v_1	1	1	1	0	0	0	0
v_2	0	0	1	1	1	0	0
v_3	0	0	0	0	1	1	1
v_4	0	0	0	0	0	0	0

$k_{G_2} = 3(1) + 3(2) + 3(3) = 18$

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
v_1	1	1	0	0	0	0	0
v_2	0	0	1	1	0	0	0
v_3	0	0	0	0	1	1	0
v_4	0	0	0	0	0	0	1

$k_{G_2} = 2(1) + 2(2) + 2(3) + 1(10) = 22$

In the case of G_2, the $(3, 0)$ matrix realizes the cheapest game.

Costs on Answers with Restricted Questions

Now that we have looked at creating cheap guessing games when cost functions are applied to the answer vectors and the number of questions asked did not effect the cost, we will now look at guessing games with the cost function applied to the answer vectors and a restricted number of questions when $c(\mathcal{V}) = (1, 1, ..., 1)$ only. To create the cheapest guessing game matrices for games that have these additional characteristics, we must introduce a technique that was used throughout this topic.

Definition 6. *An $m \times n$ $(1, 2, 3)$ matrix $[G]$ has the following form:*

v_1	1	...	...	...	1	...	1	...	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_i	...	1	...	1	...	1	...	...	0
v_{i+1}	1	1	0	0	0	0	0	...	0
v_{i+2}	0	0	1	1	0	0	0	...	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_{m-1}	0	...	0	0	0	0	1	1	0
v_m	0	...	0	0	0	0	0	0	1

such that $|v_1|, ..., |v_i| = 3$, $|v_{i+1}|, ..., |v_{m-1}| = 2, |v_m| = 1$ and $d(G) \geq 3$.

Now that we have introduced a category of matrices that we will be using, we will now introduce two designs that we will use to show when these matrices can be constructed.

Definition 7. *A **Steiner Triple System** on u points (STS(u))is a set of triples from a set X containing u elements, called blocks, in X such that each pair of the varieties of X occur in exactly one block.*

Definition 8. *The **replication number** r of a Steiner Triple System is the number of times each variety appears in a block such that $r = \frac{n-1}{2}$. [2]*

Definition 9. *Let K be a set of natural numbers. A $(n, K, 1)$ **Pairwise Balanced Design** (PBD) is a pair (N, B) such that N is an $n-$ set of varieties and B is a set of subsets of N, called blocks, such that for each $b \in B$, $b \subset N$ and $|b| \in K$. In addition, each pair of the n elements of N appear in exactly 1 block in B.*

Definition 10. *Let D be a Steiner Triple System or a Pairwise Balanced Design. Let $P = \{p_1, p_2, p_3, ..., p_u\}$ be the set of u varieties of D and let $B = \{b_1, b_2, b_3, ..., b_v\}$ be the set of all blocks of D. The **incidence matrix** of D is a $|B| \times u$ binary matrix in which each entry D_{ij} is a 1 if block b_i contains variety p_j and a 0 if block b_i does not contain variety p_j..*

Example. Let $B = \{\{1,2,4\},\{2,3,5\},\{3,4,6\},\{4,5,7\},\{5,6,1\}.\{6,7,2\},\{7,1,3\}\}$. Then B is an STS(7) with $r = 3$. The incidence matrix of B is shown in Figure 8.

	1	2	3	4	5	6	7
b_1	1	1	0	1	0	0	0
b_2	0	1	1	0	1	0	0
b_3	0	0	1	1	0	1	0
b_4	0	0	0	1	1	0	1
b_5	1	0	0	0	1	1	0
b_6	0	1	0	0	0	1	1
b_7	1	0	1	0	0	0	1

Figure 8: The incidence matrix of B.

These incidence matrices will provide the structure of the matrices that will represent our guessing games.

Lemma 4. *For every $(m, c, 1)$ offline game G with n questions such that $n \equiv 1$ (mod 6) or $n \equiv 3$ (mod 6) there exists an $m \times n$ guessing game matrix $[G]$ such that $[G]$ is a valid $(1, 2, 3)$ matrix.*

Proof. For each $n \equiv 1$ (mod 6) or $n \equiv 3$ (mod 6) there exists a Steiner Triple System S of order n with varieties $\{1, 2, 3, ..., n-1, n\}$, a set of blocks B such that $|B| = \frac{n(n-1)}{6}$, and replication number r such that $r = \frac{n-1}{2}$ [4]. Let $t = \{\{a, b\}|\{a, b, n\} \in B\}$. We will replace all the blocks in B that originally contained n with t. We are then left with $\frac{(n(n-1))}{6} - r$ blocks of weight 3 and $r = \frac{n-1}{2}$ of weight 2.

After constructing our blocks of weight 2, we add a single block of weight 1 to S. This block will contain the single variety n. Since we have altered S, we are now left with a new design, S'.

Figure 9 shows the incidence matrix of our modified $STS(n)$, $[S']$.

v_1	1	...	...	...	1	...	1	...	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_i	...	1	...	1	...	1	...	...	0
v_{i+1}	1	1	0	0	0	0	0	...	0
v_{i+2}	0	0	1	1	0	0	0	...	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_{m-1}	0	...	0	0	0	0	1	1	0
v_m	0	...	0	0	0	0	0	0	1

Figure 9: The incidence matrix $[S']$.

We can see that the incidence matrix above contains one row with a weight of 1, $\frac{n-1}{2}$ rows with weight 2, while the remaining rows have weight 3. Since we have shown that $[S']$ contains the correct number of rows with the same weights as a $(1, 2, 3)$ matrix, we will now show that $d(S') \geq 3$.

Let v_i and v_j be the i-th and j-th rows of $[S']$ such that $|v_i| = 3$ and $|v_j| = 3$. Since every pair of the n varieties of S appear in exactly 1 block in B, it follows that rows v_i and v_j can only have up to one 1 in the same column. Thus, it follows that $d(v_i, v_j) \geq 4$.

Let v_i and v_j be two rows of $[S']$ such that $|v_i| = 3$ and $|v_j| = 2$. Since every pair of the n varieties in N appear in exactly 1 block in B, it follows that rows v_i and v_j can only have up to one 1 in the same column. Thus, it follows that $d(v_i, v_j) \geq 3$.

Let v_i and v_j be two rows of $[S']$ such that $|v_i| = 3$ and $|v_j| = 1$. Since the row v_j contains only one 1 in the column that represents the variety n and no other row contains a 1 in that column, it follows that $d(v_i, v_j) = 4$.

Let v_i and v_j be two rows of $[S']$ such that $|v_i| = 2$ and $|v_j| = 2$. By construction, these two rows to two blocks b_j and b_i which both contained n in S. Since every pair of the n varieties of S appear in exactly 1 block, it follows that b_i and b_j cannot shareany point other than n. Thus, the column of $[S']$ that represents n is the only column in which the rows representing the blocks in t may all contain a 1. Since the rows of $[S']$ that have weight 2 were constructed from the blocks in t, it follows that none of these rows will contain a 1 in the same column as another row of weight 2. Thus, it follows that $d(v_i, v_j) = 4$.

Thus, $d([S']) \geq 3$. Since we have shown that $[S']$ contains one row with a weight of 1, $\frac{n-1}{2}$ rows with weight 2, while the remaining rows have weight 3 and $d(S') \geq 3$, it follows that $[S']$ is a valid $(1, 2, 3)$ guessing game matrix. □

Lemma 5. *For every (m, c, ℓ) offline game G with n questions such that $n \equiv 5 \pmod 6$ there exists a valid guessing game matrix $[G]$ such that $[G]$ is a $(1, 2, 3)$ matrix.*

Proof. For $n \equiv 5 \pmod 6$, there exists a Pairwise Balanced Design D with n varieties such that $N = \{1, 2, 3, ..., n-1, n\}$, $K = \{3, 5\}$, and D has exactly one block of size 5 while the rest of the blocks have size 3 [4]. Thus, it follows that D will contain $\frac{\binom{n}{2}-\binom{5}{2}}{\binom{3}{2}}$ blocks of weight 3. Let $\{a, b, c, d, e\}$ be the block of weight 5 in D. We remove the variety a from all blocks. We observe that the variety a is contained in $\frac{n-5}{2}$ blocks of size 3 since each pair of varieties appears in exactly one block, and in the size 5 block, a is already paired with 4 other varieties and cannot be paired with itself. Thus, removing a from all blocks leaves us with $\frac{n-5}{2}$ blocks of size 2 and a block of size 4. We are able to decompose this block of size 4 into a block of size 3 and a block with 3 pairs all sharing one variety,

which we will call B_1, as shown in Figure 10.

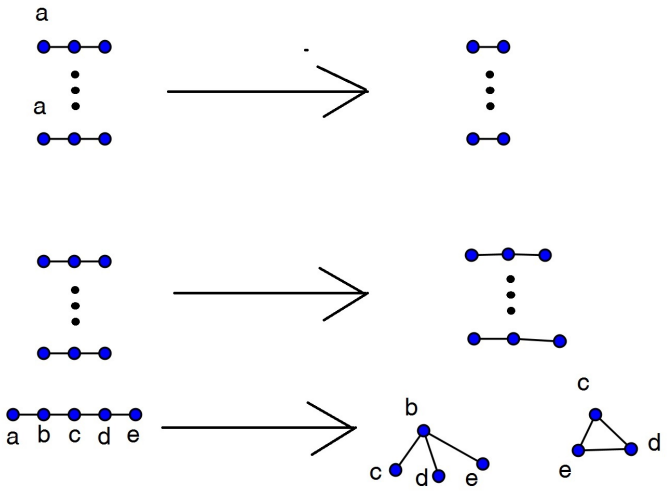

Figure 10: Replacing blocks containing a (right) with t (left), blocks of size 3 *not* containing a (left) remain the same (right) and the block of size 5 (left) is replaced with a block with 3 pairs all sharing one variety and a block of size 3.

From the block of size 3 and B_1 that we constructed from the block of size 4, we can isolate the pairs $\{b, c\}$ and $\{d, e\}$ as shown in Figure 10 from each structure as shown the diagram below. Thus, we are left with $\frac{n-5}{2} + 2 = \frac{n-1}{2}$ blocks of size 2. Therefore, the resulting structure contains $\frac{\binom{n}{2}-\binom{5}{2}}{\binom{3}{2}} - \frac{n-5}{2}$ blocks of size 3 with $n-1$ varieties.

Let T be the set of all pairs in $N \backslash \{a\}$. We construct the blocks of size 2 in our game matrix such that each of these vectors contain a pair in T. Therefore, we will construct $\frac{n-1}{2}$ blocks of size 2 with $n-1$ varieties.

After constructing our blocks of size 2, we add a single block of size 1 containing the single variety a. The resulting structure D' that has been constructed from D will have an incidence matrix $[D']$ shown in Figure 11.

v_1	1	...	...	...	1	...	1	...	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_i	...	1	...	1	...	1	...	...	0
v_{i+1}	1	1	0	0	0	0	0	...	0
v_{i+2}	0	0	1	1	0	0	0	...	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
v_{m-1}	0	...	0	0	0	0	1	1	0
v_m	0	...	0	0	0	0	0	0	1

Figure 11: The incidence matrix $[D']$.

Similarly to Lemma 4, by using the fact that every pair in N appears in exactly one block of D, it follows that $d([D']) \geq 3$. Since we have shown that $[D']$ contains one row with a weight of 1, $\frac{n-1}{2}$ rows with weight 2 which are disjoint, while the remaining rows have weight 3 and $d(D') \geq 3$, it follows that $[D']$ is a valid $(1,2,3)$ guessing game matrix. □

Definition 11. *For an (m,c,ℓ) offline game G with n questions, a punctured $(1,2,3)$ matrix is a guessing game matrix $[G]$ with $|v_1|,...,|v_{\lfloor \frac{n}{2} \rfloor}| = 2$ and $|v_{\lfloor \frac{n}{2} \rfloor} + 1|,...,|v_m| = 3$ and $d(G) \geq 3$.*

Lemma 6. *For every (m,c,ℓ) offline guessing game G with n questions whose game matrix is a $(1,2,3)$ matrix $[G]$, when n is odd, there exists a $(n-1,c,\ell)$ game G' with a punctured $(1,2,3)$ game matrix $[G']$ such that $[G']$ may be constructed from $[G]$.*

Proof. Let G be a $(m,c,1)$ offline guessing game with n questions such that n is odd and let the guessing game matrix $[G]$ be a $(1,2,3)$ matrix. Remove the column containing a single 1 from $[G]$ to create the matrix $[G']$. This will not reduce the minimum distance of $[G']$ since we are removing a column with a single 1 such that this 1 is also the only 1 in its row. Thus G' is a valid $(n-1,c,1)$ guessing game. Since $[G']$ is the guessing game matrix that represents G' and contains vectors such that $|v_2|,...,|v_{\lfloor \frac{n}{2} \rfloor}| = 2$,and$|v_{\lfloor \frac{n}{2} \rfloor + 1}|,...,|v_m| = 3$, it follows that $[G']$ is a punctured $(1,2,3)$ matrix. □

Theorem 7. *For every $(m,c,1)$ guessing game G with n questions such that $c(\mathcal{V}) = (1,1,...,1)$, if $[G]$ is a $(1,2,3)$ matrix, then $[G]$ is the guessing game matrix of the cheapest possible game.*

Proof. For every $(m,c,1)$ guessing game G with n questions and game matrix $[G]$ such that $[G]$ is a $(1,2,3)$ matrix and $c(\mathcal{V}) = (1,...,1)$. We will partition $[G]$ into two sets of vectors, A and B, such that for all $v_a \in A$, $|v_a| = 3$ and for all $v_b \in B$, $|v_b| \in \{1,2\}$. This partitioning is shown in Figure 12.

$A =$

v_1	1	...	...	...	1	...	1	...	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
v_i	...	1	...	1	...	1	...	...	0

$B =$

v_{i+1}	1	1	0	0	0	0	0	...	0
v_{i+2}	0	0	1	1	0	0	0	...	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
v_{m-1}	0	...	0	0	0	0	1	1	0
v_m	0	...	0	0	0	0	0	0	1

Figure 12: The matrix $[G]$ partitioned into A and B.

We will first show that partition B cannot be reduced further. We can see that B is a $(2,1)$ construction as defined in Section 1. Since $c(\mathcal{V}) = (1,...,1)$, by Theorem 1, it follows that a $(2,1)$ matrix realizes the cheapest game for partition B. Thus B cannot be reduced further. Since $d(G) \geq 3$, it follows

that no two vectors v_j and v_k in B may both contain a 1 in the same column. Since G is restricted to having exactly n questions, it follows that the partition B contains exactly $\lfloor \frac{n}{2} \rfloor$ vectors of weight 2.

Partition A contains only vectors of weight 3. We claim that this construction of A cannot be reduced. Suppose A contains a vector v_a such that $|v_a| = 2$. For each of the $n-1$ columns of $[G]$ that do not contain a single 1, there exists a row of weight 2 that contains a 1 in that column. Therefore, it follows that there exists a vector $v_b \in B$ such that that $d(v_a, v_b) < 3$. Thus, there cannot exist such a vector $v_a \in A$. Therefore, it follows that for all vectors $v_i \in A$, $|v_i| \geq 3$.

We now claim that, in order to construct the cheapest game, it must be the case that for all $v_a \in A$, $|v_a| = 3$. Let $k_i = |v_1| + ... + |v_i| + ... + |v_m|$ and $k_j = |v_1| + ... + |v_j| + ... + |v_m|$ such that $|v_i| \geq 4$ and all other vectors have weight 3. It then follows that

$$k_i - k_j \geq 1.$$

Therefore, $k_i - k_j \geq 0$ so $k_i \geq k_j$ thus constructing vectors such that $|v_1|, ..., |v_m| = 3$ will decrease or not change the total cost of G. Thus, this vector construction will give us the cheapest possible guessing game.

Since we have seen that partitions A and B cannot be reduced further it follows that a $(1, 2, 3)$ matrix will give the guessing game matrix of the cheapest game for all (m, c, ℓ) games with n questions when $c = (1, ..., 1)$. □

4 Summary and Future Work

In this paper, we have shown how to minimize the total cost of any offline guessing game with an unrestricted number of questions for *any* cost function when 1 lie is allowed. We have also shown how to minimize the total cost of any offline guessing game with a restricted number of questions for the cost function $c(\mathcal{V}) = (1, 1, ..., 1)$.

Table 1 is a table of the minimum cost of $(m, c, 1)$ guessing games with n questions when $c(\mathcal{V}) = (1, 1, ..., 1)$.

$m\backslash n$	**6**	**7**	**8**	**9**	**10**
3	6	5	6	5	6
4	9	7	8	7	8
5	12	10	11	9	10
6	15	13	14	12	13
7	18	16	17	15	16
8	x	19	20	18	19
9	x	22	23	21	22
10	x	25	26	24	25
11	x	28	29	27	28
12	x	x	32	30	31
13	x	x	x	33	34
14	x	x	x	36	37
15	x	x	x	39	40

Table 1: The minimum cost of $(m, c, 1)$ games with n questions.

For others who are interested in investigating guessing games with cost functions, the following are some open ended questions that could be addressed in the future:

- We have shown the minimum cost for offline games that allow for only 1 lie. So, a natural direction for future research is to minimize the total cost of offline (m, c, ℓ) games where $\ell > 1$.

- We have shown the minimum cost of offline games with restricted number of questions for one cost function. Future work could investigate how to minimize the cost of these restricted games with other cost functions.

- We have shown the minimum cost of offline guessing games with cost functions, so future work could investigate how to minimize the total cost of *online* guessing games with cost functions.

5 Acknowledgments

I would like to thank the Student Summer Scholars (S3) program and at Grand Valley State University and all of the program's wonderful donors for funding this research. I would also like to thank my research advisor, Dr. David Clark, for his guidance, advice, and positivity throughout this project.

References

[1] A.E. Brouwer, E. Andries, L.B. Shearer, and N. I. A. Sloane, *A new table of constant weight codes*, IEEE Trans Inform Theory (1990), 1–19.

[2] R. Brualdi, *Introductory Combinatorics*, Columbia Press (2009).

[3] C. Deppe, *Coding with feedback and searching with lies*, Entropy, Search, Complexity. (2007), 27–70.

[4] C. Lindner, *Design Theory*, Discrete Mathematics (1997).