Minimizing the Cost of Guessing Games

David Clark[a], *Lindsay Czap*[*b]

[a] *Department of Mathematics, Grand Valley State University, Allendale, MI*
[b] *Department of Mathematics, Middle Tennessee State University, Murfreesboro, TN*

Students: czapl@mail.gvsu.edu, lnc4a@mtmail.mtsu.edu
Mentor: clarkdav@gvsu.edu

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. In this paper, we study guessing games with lies and costs. In particular, the Responder is allowed to lie in one answer, and the Questioner is charged a cost based on the content of each question. Guessing games with lies are closely linked to error correcting codes, which are mathematical objects that allow us to detect an error in received information 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 additionally seek to minimize the total cost of our games. We will provide explicit constructions, for any cost function, for games with the minimum possible cost and an unlimited number of questions. We also find minimum cost games for games with a restricted number of questions and a constant cost function.

KEYWORDS
Ulam's Game; Guessing Games With Lies; Error Correcting Codes; Pairwise Balanced Designs; Steiner Triple Systems

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. In this paper, we will focus on a different type of guessing game: In an *offline* guessing game, the questioner must ask all of the questions at once and then receive all of the responses at once afterwards. 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.

In our offline guessing games, the two players decide on a range of m integers starting at 1. The first player, the "Responder," secretly chooses one of those integers x. The second player, the "Questioner," then constructs yes-or-no questions of the form "Is $x \in S$?" where $S \subseteq \{1, 2, ..., m\}$. Since these games are offline, all of the questions that the Questioner wishes to ask the Responder will be asked at once and then all of the answers will be received at once.

Example 1. The following is an example of an offline guessing game with $m = 4$ possible answers.

Questioner:

Question 1: "Is your number in $\{3, 4\}$?"
Question 2: "Is your number in $\{2, 4\}$?"
Question 3: "Is your number in $\{2, 3\}$?"
Question 4: "Is your number in $\{2, 4\}$?"
Question 5: "Is your number in $\{3, 4\}$?"

Responder: Responses: "Yes", "No", "Yes", "No", "Yes" (in order).

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 $\{1, 2\}$, leaving only $\{3, 4\}$ as possibilities. A response of "no" to Question 2 eliminates the possibility of 4 being the correct number, leaving only 3 as a possibility. Note that Question 2 also eliminates 2 as a possibility, but the Responder's answer to Question 1 already told us this. The remaining answers are no longer necessary, but they serve to confirm this result (and will be useful later, when we begin to discuss guessing games with lies). Thus, the Questioner knows that the Responder chose $x = 3$ as their integer.

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. We will also add the idea of *cost* to a guessing game: The Questioner will be charged a certain cost whenever they ask a question. In the following sections, we will give precise definitions and background for each of these additional twists.

BACKGROUND

In this section, we give background and definitions, leading up to a precise definition of guessing games with lies and cost functions.

Guessing games with lies have been extensively studied. The idea of a "searching game" with a lie was suggested by Ulam in his autobiography[1], which in turn spawned a large and active field of research. In general, work in this area has focused on minimizing the number of questions needed to determine the Responder's number correctly. In particular, Spencer[2] focused on solving "Ulam's game" with 1,000,000 numbers and 1 lie, which turns out to be the equivalent of the "20 questions" game with 1 lie permitted. Offline guessing games with lies turn out to be equivalent to important objects in mathematics called *error-correcting codes*. See[3,4] for some excellent surveys of the literature on guessing games with lies.

Our particular focus in this paper will be adding the idea of a "cost function" to offline guessing games with lies. A guessing game G with cost function c is an offline guessing game in which each of the possible answers i is assigned a "cost" $c(i)$. For each question, the Questioner will be "charged" the cost of all of the 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 costs of 1 and 3. This generalizes the idea of minimizing the number of questions required, and allows us to measure the "efficiency" of a game in a more general way.

Representations of guessing games

For our offline guessing games, we will represent the answers to each of our questions using binary vectors. A binary vector is an ordered list of 1's and 0's $(x_1, x_2, \ldots, x_n)$ where $x_i = 0$ if the Responder's answer to the ith question is "no", and $x_i = 1$ if the answer is yes. For example, if we asked 5 consecutive questions and their respective answers were "yes", "yes","no," "yes","no", we would represent these responses with the binary vector $(1, 1, 0, 1, 0)$. We will often use the shorthand notation: 11010.

We can think of an answer vector as the "fingerprint" of the Responder's secret number. It contains all of the information necessary to determine the secret number. In particular, in every guessing game, every secret number must have a different answer vector or else it would be impossible to distinguish between some of the numbers.

Definition 2. For a given natural number m, a *cost function* c is a function $c : \{1, 2, \ldots, m\} \to \mathbb{R}^+$ with $0 < c(1) \leq c(2) \leq ... \leq c(m)$. For simplicity, we often represent c as a vector: $c = (c(1), c(2), ..., c(m))$.

We can now define the fundamental object in this paper:

Definition 3. An (m, c, ℓ) *offline guessing game G with n questions* is an ordered list of n questions of the form "Is your number in the set S_i?", where $S_i \subseteq \{1, 2, \ldots, m\}$, together with a cost function $c = (c(1), c(2), \ldots, c(m))$. The questions are arranged so that the Questioner can determine the Responder's secret number, even if the Responder is allowed to lie in at most ℓ answers.

We will often omit the word "offline", since we will only be working with offline guessing games in this paper. Note that we have not yet indicated *how* the questions may be arranged to let the Questioner determine the Responder's secret number in the face of lies. We will address this in the following sections.

Next, we give a more precise definition of the "answer vectors" described at the beginning of this section. Here, let $\mathbb{F}_2^n$ be the space of all vectors of length n over the finite field $\mathbb{F}_2$, that is, binary vectors of length n.

Definition 4. Given an (m, c, ℓ) guessing game G with n questions, the *answer vector for* i, denoted $\mathbf{v}_i$, is the binary vector $\mathbf{v}_i \in \mathbb{F}_2^n$ consisting of the Responder's truthful answers to the n questions, in order, assuming that the Responder's secret number is i. The *answer vector set of* G, denoted $\mathcal{V}_G$, is the set $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\} \subseteq \mathbb{F}_2^n$ of answer vectors for $i = 1, 2, \ldots, m$.

Definition 5. Given an (m, c, ℓ) game G with n questions, the *guessing game matrix of* G is an $m \times n$ binary matrix denoted $[G]$ in which row j corresponds to vector $\mathbf{v}_j \in \mathcal{V}_G$. That is, the jth row of $[G]$ is the list of the Responder's truthful answers to the n questions in order, assuming that the Responder's secret number is j.

Answer vectors and guessing game matrices are the main ways in which we will study guessing games in this paper. Because the rows of a guessing game matrix are the same as the answer vectors, we will often refer to the rows of a matrix as "vectors."

Example 6. There are many different vector sets that could be used to construct a $(4, c, 1)$ game. Consider the set $\mathcal{V}_G = \{00000, 01110, 10101, 11011\} \subseteq \mathbb{F}_2^5$. This is a set of 4 answer vectors of length 5 for the numbers $\{1, 2, 3, 4\}$. This could be the answer vector set for a $(4, c, 1)$ guessing game with 5 questions. In particular, we have $\mathbf{v}_1 = 00000$, $\mathbf{v}_2 = 01110$, $\mathbf{v}_3 = 10101$, and $\mathbf{v}_4 = 11011$. Notice that $\mathbf{v}_3$ corresponds to the answers given by the Responder in Example 1.

Figure 1 shows the game matrix $[M]$ for a $(4, c, 1)$ game M that uses the answer vector set $\mathcal{V}_G$.

$[M] =$

	q_1	q_2	q_3	q_4	q_5
1	0	0	0	0	0
2	0	1	1	1	0
3	1	0	1	0	1
4	1	1	0	1	1

Figure 1. A possible game matrix, $[M]$, for a $(4, c, 1)$ game.

It is possible to determine a game's questions entirely from its guessing game matrix. Consider the column labeled q_1 in **Figure 1**. We say that this column is the question vector $q_1 = (0, 0, 1, 1)$. We know that, if responding truthfully, the Responder will answer "yes" to this 1st question if and only if there is a 1 in the row of $[M]$ corresponding to their secret number. Therefore, q_1 would be the first question in the game G and would be asked as: "Is your number in $\{3, 4\}$?" because these are the answers whose answer vector have a 1 in column 1 of $[M]$. This is indeed the first question asked in Example 1.

We will often treat a guessing game not as a list of questions, but rather as a list of answer vectors. These two viewpoints are exactly equivalent.

Cost functions

We will now focus on the effect of the cost function in a guessing game. The cost function will be our measure of the "efficiency" of a game.

Definition 7. Let G be an (m, c, ℓ) guessing game. Suppose that $Q = \{p_1, p_2, \ldots, p_k\} \subseteq \{1, 2, \ldots, m\}$ is a question in G. Then the *cost of question* Q is defined to be the sum of the costs of the elements of Q, that is,

$$c(Q) = \sum_{i=1}^{k} c(p_i). \qquad \textbf{Equation 1.}$$

Furthermore, the *total cost of* G, denoted k_G, is the sum of the cost of Q for all questions Q in G, that is,

$$k_G = \sum_{\text{question } Q} c(Q). \qquad \textbf{Equation 2.}$$

Because our guessing games are offline, all of the questions will always be asked. Thus the total cost of a game is a reasonable measure of the overall "efficiency" of a game. The following gives us a convenient way to calculate the total cost of a game in terms of its answer vectors.

Definition 8. The *weight* of a vector $\mathbf{v} \in \mathbb{F}_2^n$, denoted $|\mathbf{v}|$, is the number of nonzero coordinates in $\mathbf{v}$.

Definition 9. The *total cost* k_G of an (m, c, ℓ) guessing game G with answer vector set $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_m\}$ is $k_G = c(1)|\mathbf{v}_1| + c(2)|\mathbf{v}_2| + ... + c(m)|\mathbf{v}_m|$.

Example 10. We consider a $(4, c, 1)$ game G with $c = (2, 4, 6, 8)$ and game matrix $[M]$ as shown in **Figure 1**.

Reading the columns of matrix $[M]$, we see that the questions and their costs are:

#	Question	Cost
q_1	Is your number in $\{3, 4\}$?	$6 + 8 = 14$
q_2	Is your number in $\{2, 4\}$?	$4 + 8 = 12$
q_3	Is your number in $\{2, 3\}$?	$4 + 6 = 10$
q_4	Is your number in $\{2, 4\}$?	$4 + 8 = 12$
q_5	Is your number in $\{3, 4\}$?	$6 + 8 = 14$
	Total:	62

Using Definition 9, the total cost of G is calculated by:

$$\begin{aligned} k_G &= |\mathbf{v}_1|c(1) + |\mathbf{v}_2|c(2) + |\mathbf{v}_3|c(3) + |\mathbf{v}_4|c(4) \\ &= 0c(1) + 3c(2) + 3c(3) + 4c(4) \\ &= 0 \cdot 2 + 3 \cdot 4 + 3 \cdot 6 + 4 \cdot 8 \\ &= 62. \end{aligned}$$

This demonstrates the essence of Definition 9: It is possible to count the costs of a game by using the columns of its guessing game matrix (questions), or by using its rows (answer vectors).

Error-correcting codes and lies

In order to discuss the ability for our (m, c, ℓ) guessing games to detect and correct for lying, we must first introduce some ideas from a branch of mathematics known as *coding theory*. We refer the interested reader to Huffman and Pless[5] for further definitions and examples.

Definition 11. The *Hamming Distance* between two vectors $\mathbf{x}, \mathbf{y} \in \mathbb{F}_2^n$, denoted $d(\mathbf{x}, \mathbf{y})$, is the number of coordinates in which $\mathbf{x}$ and $\mathbf{y}$ differ.

Definition 12. The *minimum distance* $d(C)$ of a set of vectors $C \subseteq \mathbb{F}_2^n$ is the smallest Hamming Distance between any two distinct vectors in C.

Example 13. Continuing to use the answer vectors from Example 6, integers 2 and 3 have answer vectors $\mathbf{v}_2 = 01110$ and $\mathbf{v}_3 = 10101$. These vectors differ in 4 positions – every position except the middle digit, where both have 1's. Thus $d(\mathbf{v}_2, \mathbf{v}_3) = 4$.

The smallest distance among any pair of vectors in $\mathcal{V}_G$ from Example 6 is 3 (between $\mathbf{v}_1$ and $\mathbf{v}_2$, as well as others), and so $d(\mathcal{V}_G) = 3$.

With these ideas in hand, we are ready to define the fundamental idea that will allow us to detect and correct lies in guessing games.

Definition 14. An (n, M, d) *error-correcting code* C (or "ECC") is a subset of $\mathbb{F}_2^n$ of size M with $d(C) = d$. We call these vectors in C *codewords*. An (n, M, d) error-correcting code can be represented by an $M \times n$ matrix denoted $[C]$ whose rows are the codewords of C.

An ECC has the capacity to detect and then correct for errors that can occur in a codeword when it is exposed to contextual factors. These factors can include noise over a digital channel, a misprint in written text, or – in the case of our guessing games – a player lying. The capacity for an ECC to detect and correct for errors is based on the minimum distance of that code.

The process by which minimum distance allows a code to correct errors is called *nearest-neighbor decoding*. For an (n, M, d) code C, a *message vector* $\mathbf{m}$ is any one codeword. It is *transmitted* through a *channel*, where errors occur in the form of bits "flipped" from 1 to 0 or vice versa. The *received message vector* $\mathbf{r}$ is then a binary vector with n coordinates that may share some entries with $\mathbf{m}$. To *decode* this message, vector $\mathbf{r}$ is compared to all of the codewords in the code. The codeword $\mathbf{c}$ that minimizes $d(\mathbf{c}, \mathbf{r})$ is chosen as the "nearest neighbor" to $\mathbf{r}$. In particular, if there are at most ℓ mistakes in $\mathbf{r}$, and the minimum distance of a code is at least $2\ell+1$, then there must be a unique codeword that is "closest" to $\mathbf{r}$ and which results in a correct decoding.

Proposition 15 (See Huffman and Pless[5]). *An (n, M, d) error-correcting code C with $d \geq 2\ell + 1$ can detect and correct up to ℓ errors.*

Example 16. We continue to work with the matrix $[M]$ shown in **Figure 1** of Example 6. We interpret the rows of $[M]$ as the vectors in an error-correcting code C. It follows that $d(C) = 3$ since the Hamming Distance between any two codewords in C is at least 3. Thus, C can detect and correct for up to $\ell = 1$ errors.

For example, if the vector $\mathbf{r} = (0, 1, 0, 1, 1)$ were received, we could compare its distance to each of the four vectors in C. The 4th vector has a distance of 1 to $\mathbf{r}$, and no other vector has a distance that is 1 or lower, and so $\mathbf{r}$ would be corrected as $(1, 1, 0, 1, 1)$.

A valid guessing game matrix $[G]$ for an (m, c, ℓ) game G with n questions can be viewed as the matrix for an $(n, m, 2\ell+1)$ error-correcting code and vice versa. The binary answer vectors in $[G]$ that are assigned to each of the m possible integers correspond to the codewords of the code represented by $[G]$. The number of lies ℓ that are allowed in G corresponds to the number of errors that the corresponding ECC can detect and correct. By this, it follows that in order for $[G]$ to be a valid guessing game matrix for an (m, c, ℓ) game G, it must be true that d($[G]$)$\geq 2\ell+1$. Thus, trying to find an (m, c, ℓ) offline guessing game G with n questions is equivalent to finding an $(n, m, 2\ell + 1)$ error-correcting code.

In the remainder of this paper, we will find $(m, c, 1)$ games with n questions by solving the equivalent problem of finding $(n, m, 3)$ error-correcting codes. However, we will have the additional requirement of minimizing the total cost of the corresponding game, a restriction which is not normally considered when constructing an error-correcting code.

RESULTS

Guessing Games with an Unrestricted Number of Questions

In this section, we will make several assumptions. First, we restrict our investigations to guessing games allowing for only one lie, that is, $\ell = 1$. Second, we place no restriction on the number of questions that the Questioner asks. Thus, for each guessing game we will specify only the number of possible answers m, and will leave the number of questions n unrestricted. Instead, we will rely on the cost function for our games to measure the efficiency of the game.

For any $(m, c, 1)$ guessing game G, either the all-zeroes vector $\mathbf{0}$ is in the set of answer vectors $\mathcal{V}_G$, or it isn't. We will show two guessing game matrix constructions that we can use to minimize the total cost of a guessing game, depending upon the inclusion or exclusion of $\mathbf{0}$.

Definition 17. An $m \times n$ $(3, 0)$ matrix $[G]_m^{(3,0)}$ is an $m \times (2m - 1)$ binary matrix with the form shown in **Figure 2**.

That is, there are $m - 1$ rows of weight 3 and a single row of weight 0 at the bottom. The rows of weight 3 are arranged in a "staircase" fashion so that each row shares a 1 in exactly one coordinate with the previous row, and in exactly one coordinate with the subsequent row.

1	1	1	1	0	0	0	0	...	0
2	0	0	1	1	1	0	0	...	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$m-1$	0	0	0	0	...	0	1	1	1
m	0	0	0	0	0	0	0	0	0

Figure 2. An $m \times n$ $(3,0)$ matrix $[G]_m^{(3,0)}$.

Definition 18. An $m \times n$ $(2,1)$ matrix $[G]_m^{(2,1)}$ is an $m \times (2m-1)$ binary matrix with the structure shown in **Figure 3**.

1	1	1	0	0	0	0	0	...	0
2	0	0	1	1	0	0	0	...	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
$m-1$	0	0	0	0	...	0	1	1	0
m	0	0	0	0	0	0	0	0	1

Figure 3. An $m \times n$ $(2,1)$ matrix $[G]_m^{(2,1)}$.

There are $m-1$ rows of weight 2 and a single row of weight 1 at the bottom. No pair of rows share nonzero coordinates.

Note that, by construction, the rows of both $(3,0)$ and $(2,1)$ matrices have a minimum distance of 3.

Lemma 19. *Let m and $c = (c(1), c(2), ..., c(m))$ be given. Among all $(m, c, 1)$ guessing games that contain an answer vector consisting of all 0's, the minimum total cost is $k_{\mathbf{0}\in\mathcal{V}} = 3c(1) + 3c(2) + ... + 3c(m-1)$. A game with guessing game matrix $[G]_m^{(3,0)}$ will attain this minimum total cost.*

Proof. Let m and $c = (c(1), c(2), ..., c(m))$ be given. By Definition 9, the $(3,0)$ matrix $[G]_m^{(3,0)}$ has total cost $c([G]_m^{(3,0)}) = 3c(1) + 3c(2) + \cdots + 3c(m-1)$. Next we will show that this is the minimum cost.

Let $[G]$ be any guessing game matrix for a $(m, c, 1)$ game with minimum total cost among all $(m, c, 1)$ games that contain a zero answer vector. By assumption, $[G]$ must contain a vector of weight 0. Because $d([G]) \geq 3$, all other rows of $[G]$ must have weight at least 3. By Definition 9 and the fact that all costs are assumed to be positive, all other rows of $[G]$ must have weight exactly 3.

Next we claim that the row of $[G]$ corresponding to answer m must be $\mathbf{0}$. (Recall that, by assumption, $c(m)$ is the largest cost.) To prove this, we let $\{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_{i-1}, \mathbf{0}, \mathbf{v}_{i+1}, \ldots, \mathbf{v}_m\}$ be a fixed set of vectors with $|\mathbf{v}_j| = 3$ for all j, minimum distance at least 3, and $\mathbf{0}$ in position corresponding to cost $c(i)$, $i \neq m$. It then follows that the cost of this game is

$$k_i = 3c(1) + 3c(2) + \cdots + 3c(i-1) + 0c(i) + 3c(i+1) + \cdots + 3c(m-1) + 3c(m).$$

Suppose now that the same vectors are rearranged to place $\mathbf{0}$ in the mth position. The cost of this new game is

$$k_m = 3c(1) + 3c(2) + \cdots + 3c(i) + \cdots + 3c(m-1) + 0c(m).$$

Thus,

$$k_i - k_m = 3c(m) - 3c(i).$$

Since $c(i) \leq c(m)$ by assumption, it then follows that $k_i - k_m \geq 0$ and therefore $k_i \geq k_m$. Thus it must be that $[G]$ contains a single vector of weight zero that represents the integer m, and all other vectors have weight 3.

Therefore by Definition 9, this minimum cost must be

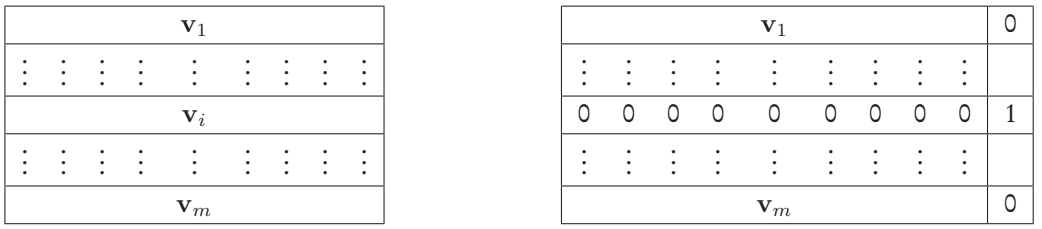
$$c([G]) = 3c(1) + 3c(2) + ... + 3c(m-1) = c([G]_m^{(3,0)}),$$

and so a $(3,0)$ matrix $[G]_m^{(3,0)}$ attains the minimum cost. □

Lemma 20. *Let m and $c = (c(1), c(2), ..., c(m))$ be given. Among all $(m, c, 1)$ guessing games that do not contain an answer vector consisting of all 0's, the minimum total cost is $k_{\mathbf{0}\notin\mathcal{V}} = 2c(1) + 2c(2) + ... + 2c(m-1) + c(m)$. A game with guessing game matrix $[G]_m^{(2,1)}$ will attain this minimum total cost.*

Proof. Let m and $c = (c(1), c(2), ..., c(m))$ be given. By Definition 9, the $(2, 1)$ matrix $[G]_m^{(2,1)}$ has total cost $c([G]_m^{(2,1)}) = 2c(1) + 2c(2) + \cdots + 2c(m-1) + c(m)$. Next we will show that this is the minimum cost.

Let $[G]$ be any guessing game matrix for a $(m, c, 1)$ game with minimum total cost among all $(m, c, 1)$ games that do not contain an answer vector of weight zero. We first claim that $[G]$ will contain a vector of weight 1. Suppose such a vector does *not* exist in $[G]$. It would follow that every vector in $[G]$ has a weight of at least 2. We will demonstrate that some vector may be reduced to a weight of 1. Suppose the set of vectors $\{\mathbf{v}_1, ..., \mathbf{v}_i, ..., \mathbf{v}_m\}$ in **Figure 4a** represent the rows of $[G]$. These rows satisfy d$([G]) \geq 3$ and all have weight of at least 2. Replace $\mathbf{v}_i$ with $\mathbf{0}$ and ask an additional question about only i, as illustrated in **Figure 4b**. We can denote this new answer vector by $\mathbf{0}|1$. It follows that $d(\mathbf{0}, \mathbf{v}_j) \geq 2$ for all $\mathbf{v}_j$, and because the single additional 1 occurs in a column of all 0's, $d(\mathbf{0}|1, \mathbf{v}_j) \geq 3$. Thus the original $\mathbf{v}_i$ can be changed to $\mathbf{0}|1$ while maintaining d$([G]) \geq 3$. This new matrix in **Figure 4b** has a strictly lower total weight than the matrix in **Figure 4a**.



(a) The rows of $[G]$ that satisfy d$([G]) \geq 3$. **(b)** The rows of $[G]$ with $\mathbf{0}$.

Figure 4. Replacing vector v_i in $[G]$ with $\mathbf{0}$.

If two vectors of weight 1 existed, then their Hamming Distance would be at most 2 and thus d$([G]) \leq 2$. Thus $[G]$ contains exactly one vector of weight 1. To ensure that $d([G]) \geq 3$, it follows that all other vectors must have a weight of at least 2. By Definition 9 and the fact that all costs are assumed to be positive, all $m-1$ rows of weight at least 2 have weight exactly 2. Thus $[G]$ will contain exactly one vector of weight 1 and $m-1$ vectors of weight 2.

Therefore by Definition 9, this minimum cost must be

$$c([G]) = 2c(1) + 2c(2) + \cdots + 2c(m-1) + c(m) = c([G]_m^{(2,1)})$$

and so a $(2, 1)$ matrix $[G]_m^{(2,1)}$ attains the minimum cost. □

We are now ready to describe the minimum-cost guessing games, where the number of questions is unrestricted.

Theorem 21. *Let m and $c = (c(1), c(2), ..., c(m))$ be given.*

- *If $c(m) < \sum_{i=1}^{m-1} c(i)$ then the minimum cost of a $(m, c, 1)$ guessing game is realized by a $(2, 1)$ matrix $[G]_m^{(2,1)}$.*
- *If $c(m) > \sum_{i=1}^{m-1} c(i)$ then the minimum cost of a $(m, c, 1)$ guessing game is realized by a $(3, 0)$ matrix $[G]_m^{(3,0)}$.*
- *If $c(m) = \sum_{i=1}^{m-1} c(i)$ then the costs of a game realized by a $(3, 0)$ matrix and a $(2, 1)$ matrix are both equal to the minimum cost.*

Proof. Let m and $c = (c(1), c(2), ..., c(m))$ be given. Let $k_{(3,0)}$ and $k_{(2,1)}$ be the total cost of an $(m, c, 1)$ guessing game realized by a $(3, 0)$ matrix and a $(2, 1)$ matrix respectively.

$[G_1] =$

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
1	1	1	1	0	0	0	0
2	0	0	1	1	1	0	0
3	0	0	0	0	1	1	1
4	0	0	0	0	0	0	0

$[G_2] =$

	q_1	q_2	q_3	q_4	q_5	q_6	q_7
1	1	1	0	0	0	0	0
2	0	0	1	1	0	0	0
3	0	0	0	0	1	1	0
4	0	0	0	0	0	0	1

Figure 5. Two possible matrices for a $(4, c, 1)$ game.

By Lemmas 19 and 20, either a $(3, 0)$ or a $(2, 1)$ matrix gives the minimum total cost of an $(m, c, 1)$ game, depending on whether or not a zero vector is present. Thus one of these two must represent the minimum possible cost of an $(m, c, 1)$ game.

Consider the case where $c(m) < \sum_{i=1}^{m-1} c(i)$. Using our assumptions and calculating the total cost of a $(2, 1)$ and $(3, 0)$ matrix, we have:

$$\begin{aligned} k_{(2,1)} &= 2c(1) + 2c(2) + \cdots + 2c(m-1) + c(m) \\ &< 2c(1) + 2c(2) + \cdots 2c(m-1) + \sum_{i=1}^{m-1} c(i) \\ &= 3c(1) + 3c(2) + ... + 3c(m-1) + 0c(m) \\ &= k_{(3,0)}. \end{aligned}$$

Thus, $c(m) < \sum_{i=1}^{m-1} c(i)$ implies $k_{(2,1)} < k_{(3,0)}$. Therefore, a $(2, 1)$ matrix gives the cheapest possible total cost.

Similarly, $c(m) > \sum_{i=1}^{m-1} c(i)$ implies $k_{(2,1)} > k_{(3,0)}$, is which case a $(3, 0)$ matrix gives the cheapest possible total cost.

If $c(m) = \sum_{i=1}^{m-1} c(i)$ then these two matrices give equal total costs. □

Example 22. Let $m = 4$. The two candidates for the cheapest $(4, c, 1)$ guessing game are given in **Figure 5**.

If $c = (1, 2, 3, 4)$, then $c(4) = 4 < 1 + 2 + 3 = \sum_{i=1}^{3} c(i)$. Theorem 21 predicts that a $(2, 1)$ matrix will be cheapest. We calculate: $c([G_1]) = 3(1) + 3(2) + 3(3) = 18$ and $c([G_2]) = 2(1) + 2(2) + 2(3) + 1(4) = 16$. Thus the $(2, 1)$ matrix does indeed realize the cheapest game.

If instead $c = (1, 2, 3, 10)$, then $c(4) = 10 > 1 + 2 + 3 = \sum_{i=1}^{3} c(i)$. Theorem 21 predicts that a $(3, 0)$ matrix will be cheapest. We calculate: $([G_1]) = 3(1) + 3(2) + 3(3) = 18$ and $([G_2]) = 2(1) + 2(2) + 2(3) + 1(10) = 22$. Thus the $(3, 0)$ matrix is cheaper in this case.

Guessing Games with a Restricted Number of Questions

In the previous section, we saw that the cheapest guessing games required us to create a very large number of questions. In particular, any guessing game with m possible answers requires $2m - 1$ questions. We will now look at guessing games where the number of questions are restricted. As a simplification, we will only consider the cost function $c = (1, 1, \ldots, 1)$. We begin, as we did in the previous section, by defining a special type of matrix that will be used throughout our study of this topic.

Definition 23. An $m \times n$ $(1, 2, 3)$ matrix $[G]$ is a matrix with minimum distance 3 which has the form shown in **Figure 6**. Here, [A] is a matrix with rows of weight 3, and [B] is a matrix with $\lfloor \frac{n-1}{2} \rfloor$ rows of weight 2.

Note that, to guarantee a minimum distance of at least 3, no row of weight 2 or 3 can have a 1 in the rightmost column. In addition, $\lfloor \frac{n-1}{2} \rfloor$ is the largest number of rows of weight 2 that can be contained in a matrix with $n-1$ columns and minimum distance at least 3. No two rows with weight 2 can share a 1 in any position.

Next, we will show that $(1, 2, 3)$ matrices (when they exist) give the lowest possible cost.

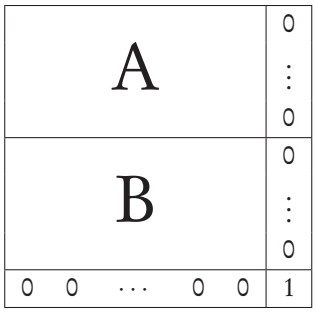


Figure 6. A $(1, 2, 3)$ matrix, [G].

Theorem 24. *Let* $c = (1, 1, \ldots, 1)$ *and let* $n \geq 3$. *If a* $m \times n$ $(1, 2, 3)$ *matrix* $[M]$ *exists, then* $[M]$ *attains the minimum total cost for an* $(m, c, 1)$ *guessing game with exactly* n *questions. This minimum cost is exactly* $3(m - \lfloor \frac{n-1}{2} \rfloor - 1) + 2\lfloor \frac{n-1}{2} \rfloor + 1 = 3m - \lfloor \frac{n-1}{2} \rfloor - 2$.

Proof. Let m and n be parameters for an $(m, c, 1)$ guessing game with exactly n questions, and assume an $m \times n$ $(1, 2, 3)$ matrix $[G]$ exists. Because the cost function is fixed at $c = (1, 1, \ldots, 1)$, the total cost of a $(m, c, 1)$ guessing game with game matrix $[G]$ is exactly the number of 1's in $[G]$. Thus our goal is to determine the minimum total number of 1's among all $m \times n$ binary matrices $[G]$ which have a minimum distance of at least 3.

Let $[G]$ be a $m \times n$ guessing game matrix with $d([G]) \geq 3$ that attains the minimum possible number of 1's.

We first claim that $[G]$ does not contain a zero vector. If $[G]$ did contain a zero vector, then to ensure $d([G]) \geq 3$, all other vectors must have weight at least 3. Note that $[M]$ must contain at least one row of weight 2, and so this guarantees that $[G]$ has at least as many 1's as $[M]$. Thus we may assume that $[G]$ does not contain a zero vector.

Next, we claim that $[G]$ must contain a single vector of weight 1. Certainly $[G]$ cannot have two or more vectors of weight 1 while maintaining $d([G]) \geq 3$, and so it can contain either 1 or 0 such vectors. Suppose it has none. Then even if $[G]$ has the maximum possible number of rows of weight 2, $[G]$ must have at least one more vector of weight 3 or larger compared to $[M]$. Thus $[G]$ would not have lower weight than $[M]$.

Thus we may assume that $[G]$ has no zero vector and has exactly one vector of weight 1. Without loss of generality, assume that this single 1 appears in column n. To ensure $d([G]) \geq 3$, any rows of weight 2 may not have a 1 in column n. Thus at most $\lfloor \frac{n-1}{2} \rfloor$ of the remaining rows may have weight 2, and the rest must have weight at least 3. A $(1, 2, 3)$ matrix meets these bounds exactly, and so it must represent the minimum possible cost.

Note that the cost of an $m \times n$ $(1, 2, 3)$ matrix is $3(m - \lfloor \frac{n-1}{2} \rfloor - 1) + 2\lfloor \frac{n-1}{2} \rfloor + 1$, because $\lfloor \frac{n-1}{2} \rfloor$ of the m rows have weight 2, one has weight 1, and the rest of the m rows have weight 3. □

In the following work, we will prove that $(1, 2, 3)$ matrices exist in certain cases. To do so requires some concepts from a branch of math known as Design Theory. The interested reader is referred to Lindner[6] for more information beyond that presented below.

Definition 25. Let $X = \{1, 2, \ldots, u\}$ for $u \geq 3$ (these elements are called *varieties*), together with a set B of subsets of X of size 3 (called *blocks* or *triples*), such that every pair of varieties in X occurs in exactly one block of B. The pair (X, B) is a *Steiner Triple System* of u varieties denoted STS(u).

Proposition 26 (see[6])**.** *An STS(u) exists if and only if* $u \equiv 1$ (*mod* 6) *or* $u \equiv 3$ (*mod* 6).

Definition 27. The *replication number* r of an STS(u) is the number of blocks that contain a fixed variety.

It can be shown[7], for an STS(u), that the replication number r satisfies $r = \frac{u-1}{2}$, regardless of the variety chosen. Similarly, the number of blocks is exactly $b = \frac{u(u-1)}{6}$.

Steiner Triple Systems will provide the basis for constructing $(1, 2, 3)$ matrices. However, they exist only for a limited number of varieties. Thus we consider an alternative structure to cover some of the remaining cases.

Definition 28. Let $X = \{1, 2, \ldots, u\}$ for $u \geq 3$ (these elements are called *varieties*). Let K be a non-empty set of natural numbers. Let B be a set of non-empty subsets of X called *blocks*, such that for each $b \in B$, $|b| \in K$. In addition, each pair of the u elements of X must appear in exactly one block of B. The pair (X, B) is a $(u, K, 1)$ *Pairwise Balanced Design*, also denoted by $(u, K, 1)$-PBD.

Note that the key difference between an $STS(u)$ and a $PBD(u, K, 1)$ is that the sizes of the blocks in a $PBD(u, K, 1)$ may vary, while an $STS(u)$ always has blocks of size 3.

Definition 29. Let $D = (P, B)$ where $P = \{p_1, p_2, p_3, ..., p_u\}$ is a set of u varieties and $B = \{b_1, b_2, b_3, ..., b_v\}$ is a set of subsets of P. The *incidence matrix* of D is a $|B| \times |P|$ binary matrix M in which each entry M_{ij} contains a 1 if subset b_i contains variety p_j and a 0 if subset b_i does not contain variety p_j.

Example 30. Figure 7 shows the incidence matrix of an STS(7) with $X = \{1, 2, 3, 4, 5, 6, 7\}$ and

$$B = \{\{1, 2, 4\}, \{2, 3, 5\}, \{3, 4, 6\}, \{4, 5, 7\}, \{5, 6, 1\}, \{6, 7, 2\}, \{7, 1, 3\}\}.$$

	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 7. The incidence matrix of an STS(7).

Note that all parameters of the STS(u) are visible in this matrix. Each row represents one block of size 3, and so each row (that is, a block) has exactly three 1's in it. Every pair of varieties appears in exactly one block, and so every pair of columns (that is, varieties) share a 1 in exactly one position. Finally, every variety is in exactly 3 blocks, and so every column contains exactly three 1's as well.

Example 31. Figure 8 shows the incidence matrix of an $(11, \{3, 5\}, 1)$-PBD.

	1	2	3	4	5	6	7	8	9	10	11
b_1	1	1	1	1	1	0	0	0	0	0	0
b_2	1	0	0	0	0	1	1	0	0	0	0
b_3	1	0	0	0	0	0	0	1	0	0	1
b_4	1	0	0	0	0	0	0	0	1	1	0
b_5	0	1	0	0	0	1	0	1	0	0	0
b_6	0	1	0	0	0	0	1	0	1	0	0
b_7	0	1	0	0	0	0	0	0	0	1	1
b_8	0	0	1	0	0	1	0	0	1	0	0
b_9	0	0	1	0	0	0	1	0	0	0	1
b_{10}	0	0	1	0	0	0	0	1	0	1	0
b_{11}	0	0	0	1	0	1	0	0	0	1	0
b_{12}	0	0	0	1	0	0	1	1	0	0	0
b_{13}	0	0	0	1	0	0	0	0	1	0	1
b_{14}	0	0	0	0	1	1	0	0	0	0	1
b_{15}	0	0	0	0	1	0	1	0	0	1	0
b_{16}	0	0	0	0	1	0	0	1	1	0	0

Figure 8. The incidence matrix of an $(11, \{3, 5\}, 1)$-PBD.

These incidence matrices will be the starting point from which we create $(1, 2, 3)$ matrices that will ultimately represent our guessing games.

Lemma 32. *Let $n \equiv 1$ or $n \equiv 3 \pmod 6$ and $2 \leq m \leq \frac{n(n-1)}{6} + 1$. Suppose $c = (1, 1, ..., 1)$. Then there exists an $m \times n$ $(1, 2, 3)$ matrix $[G]$ which is a valid guessing game matrix for an $(m, c, 1)$ guessing game G with exactly n questions.*

Proof. Let $n \equiv 1 \pmod 6$ or $n \equiv 3 \pmod 6$. By Proposition 26 , there exists a STS(n) S with varieties $N = \{1, 2, 3, ..., n-1, n\}$, block set B such that $|B| = \frac{n(n-1)}{6}$, and replication number r such that $r = \frac{n-1}{2}$. We will use B to construct a new set of blocks.

We begin by removing variety n from all blocks in B and adding a single block $\{n\}$. More precisely, let $B' = \{b \setminus \{n\} | b \in B\} \cup \{\{n\}\}$. Then B' contains $\frac{n(n-1)}{6} - r$ (unchanged) blocks of weight 3, $r = \frac{n-1}{2}$ blocks of weight 2, and a single block of weight 1.

Let $[S']$ be the incidence matrix of $S' = (N, B')$. We will show that $d([S']) \geq 3$.

Our key observation comes from the fact that every pair of varieties in N appears in exactly one block in B. Thus in B, every pair of blocks differ by at least 4 elements. In terms of $[S]$, every pair of rows differs in at least four coordinates, meaning that the minimum distance of $[S]$ is at least 4. Because any two rows in $[S']$ of weight 3 correspond directly to blocks of S, these two rows must have distance at least 4.

Let $\mathbf{v}_i$ and $\mathbf{v}_j$ be two rows of $[S']$ such that $|\mathbf{v}_i| = 3$ and $|\mathbf{v}_j| = 2$. Based on a similar observation, vectors $\mathbf{v}_i$ and $\mathbf{v}_j$ share at most one 1 in the same coordinate. Thus, d$(\mathbf{v}_i, \mathbf{v}_j) \geq 3$.

Next consider two rows $\mathbf{v}_i$ and $\mathbf{v}_j$ of $[S']$ with $|\mathbf{v}_i| = 2$ and $|\mathbf{v}_j| = 2$. By construction, the corresponding blocks b_i and b_j of S both originally contained the variety n. Therefore b_i and b_j do not share any variety other than n. Thus, $\mathbf{v}_i$ and $\mathbf{v}_j$ share 1's in no coordinates. It follows that d$(\mathbf{v}_i, \mathbf{v}_j) = 4$. In particular, the rows of weight 2 in $[S']$ contain a single 1 in every column other than the column corresponding to variety n, which has only 0's.

Finally, because variety n has been removed from all blocks in B', the distance between any row $\mathbf{v}_i$ and the row $\mathbf{v}_n$ containing only variety n must be at least 3.

Thus, d$([S']) \geq 3$. In addition, $[S']$ contains one row with a weight of 1, $\frac{n-1}{2}$ rows with weight 2, and the remaining $\frac{n(n-1)}{6} - \frac{n-1}{2}$ rows have weight 3, for a total of $\frac{n(n-1)}{6} + 1$ rows. Thus $[S']$ forms a valid guessing game matrix for a $(\frac{n(n-1)}{6} + 1, c, 1)$ guessing game.

The construction above explains how to construct a matrix with exactly $m = \frac{n(n-1)}{6} + 1$ rows. To create a valid matrix for a game with $m < \frac{n(n-1)}{6} + 1$, we simply remove as many rows of $[S']$ as necessary, beginning with rows of weight 3. This cannot affect the minimum distance of the matrix. □

Next, we consider a case for which a Steiner Triple System does not exist.

Proposition 33 (See Lindner[6]). *For $n \equiv 5$ (mod 6), there exists an $(n, \{3, 5\}, 1)$-PBD D such that D has exactly one block of size 5 while the rest of the blocks have size 3.*

Lemma 34. *Let $n \equiv 5$ (mod 6) and $2 \leq m \leq \frac{n^2-n-2}{6}$. Suppose $c = (1, 1, \ldots, 1)$. Then there exists an $m \times n$ (1,2,3) matrix $[G]$ which is a valid guessing game matrix for an $(m, c, 1)$ guessing game G with exactly n questions.*

Proof. Let $n \equiv 5 \pmod 6$. By Proposition 33, there exists a $(n, \{3, 5\}, 1)$-PBD D with varieties $N = \{1, 2, \ldots, n\}$ and block set B that contains exactly one block of size 5 while the rest of the blocks have size 3. By counting pairs, we can determine that D contains $\frac{\binom{n}{2} - \binom{5}{2}}{\binom{3}{2}} = \frac{n^2-n-20}{6}$ blocks of weight 3. Without loss of generality, we assume that $\{n, b, c, d, e\}$ is the block of weight 5 in D.

We now create a new block set by "breaking apart" the block of size 5. This will be similar to the proof of Lemma 32, but with the added complication of the block of size 5.

Let $P = \{b \in B | n \in b \text{ and } |b| = 3\}$ be the set of triples in D containing the variety n, and notice that $|P| = \frac{n-5}{2}$. Define $P' = \{b \setminus \{n\} | b \in P\}$. Let $T = \{b \in B | n \notin b \text{ and } |b| = 3\}$ be the set of triples in D not containing the variety n. We leave T unchanged.

We construct a new set of blocks $B' = P' \cup T \cup \{\{b, c\}, \{d, e\}, \{n\}\}$. Note that $\{\{b, c\}, \{d, e\}, \{n\}\}$ is essentially a "factoring" of the block of size 5. Then the set B' contains $\frac{\binom{n}{2}-\binom{5}{2}}{\binom{3}{2}} + 3 = \frac{n^2-n-2}{6}$ blocks: $\frac{\binom{n}{2}-\binom{5}{2}}{\binom{3}{2}} - \frac{n-5}{2}$ of weight 3 from T, $\frac{n-5}{2}$ of weight 2 from P', two new blocks also of weight 2, and the new block of weight 1. We note that B' contains $\frac{n-5}{2} + 2 = \frac{n-1}{2}$ blocks of size 2, the maximum possible number of such blocks.

The resulting structure $D' = (N, B')$ that has been constructed from D will have an incidence matrix $[D']$ in the form shown in **Figure 9**.

			A				0 ⋮ 0
1	1	0	0	⋯	0	0	0
0	0	1	1	⋯	0	0	0
				⋮			0
0	0	0	0	⋯	1	1	0
0	0	0	0	⋯	0	0	1

Figure 9. The incidence matrix $[D']$.

Similarly to the proof of Lemma 32, by using the fact that every pair in N appears in exactly one block of D, it follows that d($[D']$) ≥ 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 mutually disjoint, while the remaining rows have weight 3 and d(D') ≥ 3, it follows that $[D']$ is a valid $(1, 2, 3)$ guessing game matrix for a $(\frac{n^2-n-2}{6}, c, 1)$ guessing game with n questions.

The construction above explains how to construct a matrix with exactly $m = \frac{n^2-n-2}{6}$ rows. To create a valid matrix for a game with $m < \frac{n^2-n-2}{6}$, we remove as many rows as necessary, beginning with rows of weight 3. This cannot affect the minimum distance of the matrix. □

Theorem 35. *Let $c = (1, 1, \ldots, 1)$ and let $n \geq 3$. Suppose that one of the following is true:*

- $n \equiv 1$ *or* 3 *(mod* 6*) and* $2 \leq m \leq \frac{n(n-1)}{6} + 1$
- $n \equiv 5$ *(mod* 6*) and* $2 \leq m \leq \frac{n^2-n-2}{6}$

Then a $(1, 2, 3)$ matrix exists and provides the cheapest possible cost for an $(m, c, 1)$ guessing game with exactly n questions.

Proof. Under the assumptions on n, an $m \times n$ $(1, 2, 3)$ matrix $[M]$ exists by Lemmas 32 and 34. Note that if m is not as large as possible, then some rows of weight 3 must be removed from the corresponding $(1, 2, 3)$ matrix, as described in the proof of the respective Lemma. By Theorem 24, $[M]$ gives the cheapest possible cost for an $(m, c, 1)$ guessing game with exactly n questions. □

We note that Theorem 35 covers the interesting case where $m \leq \frac{n-1}{2} + 1$, in which case a $(1, 2, 3)$ matrix $[M]$ contains only rows of weight 2 and 1 (because, per the definition, all rows of weight 3 must be removed before removing any rows of lower weight). In this case, $[M]$ is the same as a $(2, 1)$ matrix, although possibly with several extra columns containing all 0's (effectively, questions that are unused). Thus Theorem 35 is consistent with the conclusion of Theorem 21 applied with $c = (1, 1, \ldots, 1)$. In this case, the restriction on the number of questions has no effect.

Example 36. The following is the construction of a $(1, 2, 3)$ matrix with $m = 8$. We begin with the incidence matrix of an STS(7) from **Figure 7**. By following the construction of a $(1, 2, 3)$ matrix as described in Lemma 32, we let the variety 1 be the variety that we remove from the blocks and then replace in a block of weight 1. From this, we yield the matrix in **Figure 10**.

	1	2	3	4	5	6	7
b_1	0	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	0	0	0	0	1	1	0
b_6	0	1	0	0	0	1	1
b_7	0	0	1	0	0	0	1
b_8	1	0	0	0	0	0	0

Figure 10. The incidence matrix with variety 1 removed and a new row added.

By rearranging the columns and rows of the matrix in **Figure 10**, we obtain the $(1, 2, 3)$ matrix in **Figure 11**.

	2	4	3	7	5	6	1
b_2	1	0	1	0	1	0	0
b_3	0	1	1	0	0	1	0
b_4	0	1	0	1	1	0	0
b_6	1	0	0	1	0	1	0
b_1	1	1	0	0	0	0	0
b_7	0	0	1	1	0	0	0
b_5	0	0	0	0	1	1	0
b_8	0	0	0	0	0	0	1

Figure 11. The final $(1, 2, 3)$ matrix.

If $c = (1, 1, 1, 1, 1, 1, 1)$, this gives a $(8, c, 1)$ guessing game with exactly 7 questions and total cost $3 \cdot 8 - \lfloor \frac{8-1}{2} \rfloor - 2 = 19$.

Example 37. The following is the construction of a $(1, 2, 3)$ matrix with $m = 11$ as described in this section. We begin with the incidence matrix of an $(11, \{3, 5\}, 1)$-PBD from **Figure 8**.

By following the construction of a $(1, 2, 3)$ matrix as described in Lemma 34, we let the variety 1 be the variety that we remove from the blocks and then replace in a block of weight 1. From this, we yield the matrix seen in **Figure 12**. The blocks created from original block of weight 5 are highlighted.

	1	2	3	4	5	6	7	8	9	10	11
b_1	0	1	1	0	0	0	0	0	0	0	0
b_2	0	0	0	1	1	0	0	0	0	0	0
b_3	0	0	0	0	0	1	1	0	0	0	0
b_4	0	0	0	0	0	0	0	1	0	0	1
b_5	0	0	0	0	0	0	0	0	1	1	0
b_6	0	1	0	0	0	1	0	1	0	0	0
b_7	0	1	0	0	0	0	1	0	1	0	0
b_8	0	1	0	0	0	0	0	0	0	1	1
b_9	0	0	1	0	0	1	0	0	1	0	0
b_{10}	0	0	1	0	0	0	1	0	0	0	1
b_{11}	0	0	1	0	0	0	0	1	0	1	0
b_{12}	0	0	0	1	0	1	0	0	0	1	0
b_{13}	0	0	0	1	0	0	1	1	0	0	0
b_{14}	0	0	0	1	0	0	0	0	1	0	1
b_{15}	0	0	0	0	1	1	0	0	0	0	1
b_{16}	0	0	0	0	1	0	1	0	0	1	0
b_{17}	0	0	0	0	1	0	0	1	1	0	0
b_{18}	1	0	0	0	0	0	0	0	0	0	0

Figure 12

By rearranging the columns and rows of the matrix in **Figure 12**, we obtain the $(1, 2, 3)$ matrix in **Figure 13**.

	2	3	4	5	6	7	8	11	9	10	1
b_6	1	0	0	0	1	0	1	0	0	0	0
b_7	1	0	0	0	0	1	0	0	1	0	0
b_8	1	0	0	0	0	0	0	1	0	1	0
b_9	0	1	0	0	1	0	0	0	1	0	0
b_{10}	0	1	0	0	0	1	0	1	0	0	0
b_{11}	0	1	0	0	0	0	1	0	0	1	0
b_{12}	0	0	1	0	1	0	0	0	0	1	0
b_{13}	0	0	1	0	0	1	1	0	0	0	0
b_{14}	0	0	1	0	0	0	0	1	1	0	0
b_{15}	0	0	0	1	1	0	0	1	0	0	0
b_{16}	0	0	0	1	0	1	0	0	0	1	0
b_{17}	0	0	0	1	0	0	1	0	1	0	0
b_1	1	1	0	0	0	0	0	0	0	0	0
b_2	0	0	1	1	0	0	0	0	0	0	0
b_3	0	0	0	0	1	1	0	0	0	0	0
b_4	0	0	0	0	0	0	1	1	0	0	0
b_5	0	0	0	0	0	0	0	0	1	1	0
b_{18}	0	0	0	0	0	0	0	0	0	0	1

Figure 13. The final $(1, 2, 3)$ matrix.

ACKNOWLEDGEMENTS

Thank you to the Student Summer Scholars (S3) program and at Grand Valley State University and all of the program's wonderful donors for funding this research.

The authors also thank the anonymous referees whose suggestions helped to improve and streamline this paper.

REFERENCES

1. Ulam, S. M. (1976) *Adventures of a Mathematician*, Scribner, New York.
2. Spencer, J. (1984) Guess a number – with lying, *Math. Mag.* 57, 105–108.
3. Deppe, C. (2007) Coding with feedback and searching with lies, in entropy, search, complexity (Csiszár, I., Katona, G., Tardos, G., Eds.), *Bolyai Society Mathematical Studies* vol. 16, 27–70, Springer.
4. Pelc, A. (2002) *Searching games with errors: fifty years of coping with liars*, Theoret. Comput. Sci. 270, 71–109.
5. Huffman, W. and Pless, V. (2003) *Fundamentals of Error-Correcting Codes*, Cambridge University Press.
6. Lindner, C. (2008) *Design Theory* 2nd ed., Chapman and Hall/CRC.
7. Brualdi, R. (2009) *Introductory Combinatorics*, Pearson.

ABOUT THE STUDENT AUTHOR

Lindsay Czap completed this project while earning a bachelor's degree in mathematics from Grand Valley State University with funding from the Student Summer Scholars program. She is currently a graduate student at Middle Tennessee State University earning her master's degree in pure mathematics. She plans to pursue her doctoral studies in undergraduate mathematics education.

PRESS SUMMARY

The game "20 questions" has always been a staple for children trapped in long car rides. We explore guessing games like "20 questions" in which one player chooses a number, and the other player asks yes-or-no questions to determine the number. In our games, the responding player may lie once. By adding a cost to each question in these games, we are able to quantify their efficiency. We determine the most efficient games in two cases: When the number of questions is unlimited, and when the number of questions is limited and the costs are especially simple. In each case, we explain exactly how to play these games in the most efficient way.