Playing Guess Who with Your Kids

Ami Paz
LISN, CNRS & Paris Saclay University, France

Liat Peterfreund
LIGM, CNRS & Gustav Eiffel University, Champs-sur-Marne, France

Abstract

Guess who is a two-player search game in which each player chooses a character from a deck of 24 cards, and has to infer the other player's character by asking yes-no questions. A simple binary search strategy allows the starting player find the opponent's character by asking 5 questions only, when the opponent is honest.

Real-life observations show that in more realistic scenarios, the game is played against adversaries that do not strictly follow the rules, e.g., kids. Such players might decide to answer all questions at once, answer only part of the questions as they do not know the answers to all, and even lie occasionally. We devise strategies for such scenarios using techniques from error-correcting and erasure codes. This connects to a recent line of work on search problems on graphs and trees with unreliable auxiliary information, and could be of independent interest.

2012 ACM Subject Classification Theory of computation → Representations of games and their complexity; Theory of computation → Sorting and searching; Theory of computation → Theory and algorithms for application domains

Keywords and phrases Guess Who?, Binary Search, Error Correcting Codes, Erasure Codes

Digital Object Identifier 10.4230/LIPIcs.FUN.2022.23

Acknowledgements The authors are grateful to Yonatan, Yuval and Ari for inspiring this work, and to Ben Lee Volk for discussions on error correcting codes.

1 Introduction

Guess Who? is a two-player board game where the goal of each player is to guess the identity of its opponent chosen character. Each player has a board that includes cartoon images of 24 characters. At the beginning of the game, each player chooses one of the characters, and the goal of the other player is to guess this character. The objective of the game is to be the first player to determine which character was selected by the opponent. Players alternately ask their opponents various yes-no questions on their selected card, such as "Does your character wear glasses?", "Is your character a woman?" etc. The player then eliminates candidates based on the opponent's response by flipping the appropriate characters down. The game ends as soon as there is a player with a single character left on their board.

Nica [18] performed a very interesting game-theoretical analysis of the game. By analyzing it as a zero-sum stochastic game, he was able to show that at each round, a player with more cards left should take a "bold move", and ask a question that has the potential of eliminating many characters, while the leading player (with less characters left) should stick to a traditional binary search strategy.

In this work, we take an algorithmic perspective. We hence ignore the game-theoretic competitive considerations that paved the way in Nica's work, and instead, focus only on the underlying algorithmic search problem. To this end, we analyze a version of the game where one player is the *chooser*, which chooses a character answers questions on it, while the

other player is the *guesser*, who has to ask questions in order to correctly guess the chosen character. Thus, the focus of this paper is on strategies for the guesser. Doing so, we follow the footsteps of Knuth [15], who applied a similar approach studying the Mastermind game.

Naturally, against a well-behaved chooser, the optimal strategy will be a binary search. However, in real-world applications, such as playing Guess Who with young kids, a guesser may need to devise a strategy that yields a correct guess despite a chooser that *deviates* from the game protocol. Inspired by real-life experience, we consider several ways in which a chooser might deviate from the protocol.

A *Not-At-Time-ANswering* (NATAN) adversary is an impatient chooser, which refuses to answer one question at a time, and instead, gives the guesser a single opportunity to present all their questions, and answers them altogether. In terms of theoretical computer science, we can think of this as an asynchronous adversary, or alternatively, we can say that the guesser is *non-adaptive*. In Section 3, we show that there is a simple strategy that allows to perform a binary search even against this adversary, thus guessing in an optimal number of $\lceil \log n \rceil$ questions.

An *Answers Relatively Insecure* (ARI) adversary is a chooser that is having hard time determining if a character has some property, and thus, might answer "I do not know" and not only "Yes" or "No". This adversary, in addition to modeling a young kid, is also relevant in the context of communication in a noisy channel. Thus, it may not come as a surprise to the reader that in Section 4, we show how to overcome the indecisiveness of the adversary by using *erasure codes*.

Finally, a Valid At Large (VAL) player is a chooser that has a flexible view of the truth. Hence, this chooser might answer incorrectly to some fraction of the questions. In a game, we give this player the benefit of the doubt that they are merely mistaken, but in theoretical computer science, we would say we are dealing with a *malicious* player, or with a *Byzantine* agent. Section 5 shows that this behavior can be overcome using *error correcting codes*, with an overhead of a few questions.

The applicability of our approach in real-world scenarios is still to be proven, and an experimental follow-up work is now being considered. Regardless of this aspect, we believe that our work presents simple applications of basic theoretical computer science concepts such as binary search, erasure codes and error correcting codes, and as such can be used for introductory examples for a wide audience.

Related Work

As mentioned, Nica [18] was the first to conduct a rigorous analysis of the Guess Who game. His work stayed in the framework of the game, and used a stochastic analysis for determining when a player should make a "bold guess", that in a small probability will result in the elimination of many characters. We note that such an analysis is useless, e.g., against the NATAN adversary, as it crucially relies on the round-based nature of the game.

The ARI adversary resembles scenarios in which some of the information is missing. Handling incomplete or missing information is one of the longstanding topics in database research [16, 14, 19, 4]. In particular, the study of certain answers (answers that are true regardless of how one completes the missing entries of the queried database) has attracted lots of attention in the database community [22, 7].

The VAL adversary presents a setting of a search with partially incorrect information, that is of high research interest in recent years. This is the case, e.g., when studying search and navigation problems in graphs, where the information regarding the target's location might be unreliable [3, 12, 13, 8]. A question even more similar to ours is that of playing twenty questions game with a liar [6], which has tight connections to learning theory.

(a) Characters.[1]

	glasses	no glasses	hat	blond	back hair	brown hair	white hair	glasses and bald
Bill	0	1	0	0	0	0	0	0
Charles	0	1	0	1	0	0	0	0
Claire	1	0	1	0	0	0	0	0
Joe	1	0	0	1	0	0	0	0
Maria	0	1	1	0	0	1	0	0
Max	0	1	0	0	1	0	0	0
Sam	1	0	0	0	0	0	1	1
Susan	0	1	0	0	0	0	1	0
Tom	1	0	0	0	1	0	0	1

(b) A corresponding $\mathbf{T}_{9\times 8}$ matrix.

Figure 1 Running Example.

Guess Who was invented as a simple version of the *Bulls and Cows* game (whose commercial version is known under the name "Mastermind"), another search game where the questions and answers are more involved than the binary questions applied here. In Mastermind, players have to guess their opponent's 4-peg colorful pattern, while the opponent replying to each guess by specifying the number of correct pegs used in place, and those that are correct but misplaced. A word-based variant of Mastermind is the trendy Wordle game. While the scientific literature on Wordle is rather limited (though numerous analyses, discussions and strategies can be found online), Mastermind was studied in several occasions. In 1977, Knuth [15] devised an optimal strategy for Mastermind, in only 5 guesses. Interestingly, he analyzed the game as a game with a single guesser and ignored game-theoretic aspects yielding from the fact that the game has two players, as we do here. In recent decades, several variants [21, 1] and approaches to Mastermind were studied, most notably is the *evolutionary* approach [9, 5, 11, 10, 2].

2 Formal Framework

In this section, we set the formal settings based on our algorithmic perspective.

2.1 Characters and Traits

We denote the number of *characters* by n, and refer to each character by a unique identifier in the set $[n] := \{1, \ldots, n\}$. We denote possible (physical) *traits* (of the characters) by $[m]$. A trait can be being brunette, not wearing a hat, having a big nose and being blond etc. Since players can only ask yes-no questions, we assume that all possible questions are of the form

$q(j) :=$ Does your character possess trait j?

where $1 \leq j \leq m$.

[1] Image taken from `iSLCOLLECTIVE.com`

The information on the different characters and their traits can be represented as a binary matrix whose rows correspond with characters' identifiers, and columns with traits. Formally, the *traits' matrix*, is a binary $n \times m$ matrix $\mathbf{T}$, where $M_{i,j} = 1$ if character i possess trait j, and $M_{i,j} = 0$ if character i does not possess trait j. In this way, every row of the matrix describes a character.

▶ **Example 1.** Figure 1a presents a subset of characters of the original game and Figure 1b a corresponding matrix with traits listed as its columns.

2.2 Distinguishability

We assume that each two different characters are *distinguishable*, that is, have at least one different trait. Formally, for each $i \neq i' \in [n]$ there is $j \in [m]$ such that $\mathbf{T}[i][j] \neq \mathbf{T}[i'][j]$. This ensures that no matter which character was chosen by the chooser, the guesser can ask questions on its traits until revealing its identity.

In fact, similarly to [18], we assume even a stronger assumption – not only do every two characters are distinguishable, but also any subset of characters is distinguishable from the rest. We formalize this as follows:

▶ **Assumption 1.** *For any subset $A \subseteq [n]$ of characters, there is a trait j_A such that $A = \{i \in [n] \mid \mathbf{T}[i][j_A] = 1\}$.*

We call j_A the trait that *separates* A.

▶ **Example 2.** The trait that separates the subset consisting of Sam and Tom from the rest is "having glasses and being bald." For Sam, Tom, Joe and Claire, it is simply "having glasses".

From this point on, we only consider games for which Assumption 1 holds.

2.3 Games, Runs, and Classical Strategies

We view an n-character game, or, simply, a game, as a process in which a player, namely *the guesser*, needs to search in the n-characters' space the character chosen by the other player, namely *the chooser*. Classically, this search is done by eliminating elements from $[n]$ until reaching a single one. For simplicity, we assume that a player does not need to explicitly guess the other player's character after narrowing down the list of possible characters to one; on the other hand, a player cannot guess a character if its list is larger than one. Formally, we define a *run* ρ as a sequence:

$$q_1, a_1, I_1, \ldots, q_k, a_k, I_k$$

where for every $1 \leq i \leq k$ it holds that $I_i \subseteq [n]$, q_i and a_i are, respectively, the question and answer of round i of the game, and the following hold for every $1 \leq \ell \leq k$:

- if $q_\ell = q(j)$ and $a_\ell =$ yes then $I_\ell = \{i \in I_{\ell-1} \mid \mathbf{T}[i][j] = 1\}$;
- if $q_\ell = q(j)$ and $a_\ell =$ no then $I_\ell = \{i \in I_{\ell-1} \mid \mathbf{T}[i][j] = 0\}$,

where we set $I_0 := [n]$.

We say that $q_1, \ldots, q_k$ *defines* ρ. In addition, we say that ρ is *successful* if I_k is a singleton. If ρ is successful, we say that $q_1, \ldots, q_k$ *defines* a successful run.

▶ **Example 3.** Consider the run $q_1, a_1, I_1, q_2, a_2, I_3$ where I_0 consists of all characters that appear in Figure 1a, $q_1 = q$(hat), $a_1 =$ yes, $I_1 = \{$Claire, Maria$\}$, $q_2 = q$(brown hair), $a_2 =$ yes, and $I_3 = \{$Maria$\}$.

A *classical strategy* is a procedure that, given a run $q_1, a_1, I_1, \ldots, q_i, a_i, I_i$, either chooses the next question q_{i+1}, or terminates if the run is successful. If, for every chosen character, the procedure terminates after choosing at most k questions, we say it is a k-question strategy.

2.4 Our Gallery of Adversaries

In this paper we consider several adversaries who deviate from the standard game protocol. In order to play with each of them, we have to adjust the notion of a strategy.

Impatient NATAN. The first adversary we consider is the *impatient* adversary denoted NATAN, which answers all the questions at once. A run against such an adversary is a normal run, but the strategy is non-adaptive. Formally, a *k-question predetermined-strategy* is a set $\{q_1, \ldots, q_k\}$ of questions where $q_\ell := q(j_\ell)$, such that the following set is a *singleton*:

$$\{i \mid \forall \ell : ((a_\ell = \text{yes}) \text{ implies } \mathbf{T}[i][j_\ell] = 1 \text{ and } (a_\ell = \text{no}) \text{ implies } \mathbf{T}[i][j_\ell] = 0)\}$$

where a_ℓ is a correct answer to q_ℓ for every ℓ. That is, there is always a single character determined by answers to these questions. Alternatively, using our previous definitions, we can define *k-question predetermined-strategy* as a set $\{q_1, \ldots, q_k\}$ of questions, such that each sequence obtained by a permutation of its elements defines a successful run.

Clueless ARI. The second type of adversaries we consider is *clueless* adversaries that might occasionally say they do not know the answer. ARI is such an adversary that might answer "don't know" at most once. To deal with clueless adversaries, we slightly extend the definition of runs by allowing $a_\ell =$ don't know, and setting $I_\ell = I_{\ell-1}$ in this case. A *strategy* against a clueless adversary is defined similarly to a classical strategy, while considering the extended notion of run. It is said to be a *k-question strategy* if it terminates after at most k questions for any chosen character.

In fact, our strategy for the clueless adversary is even more elaborate, and holds against an adversary that is *both impatient and clueless*. As the reader may expect, this adversary answers all the questions at once, and might answer "don't know" on some fraction of them. We define the notion of *k-question predetermined strategy* similarly to before. Note that the alternative definition based on successful runs is valid also here by considering the extended notion of run.

Liar VAL. Finally, we consider *liar* adversaries. VAL, for example, is such an adversary that can lie at most once in a game.

A *k-question predetermined strategy* for an impatient liar that lies at most d times is a set $\{q(j_1), \ldots, q(j_k)\}$ of questions such that for there is a unique $i \in [n]$ for which

$$d_{\text{HAM}}((b_1, \ldots, b_k), v^i) \leq d$$

where d_{HAM} stands for the Hamming distance (that is, the number of entries in which two vectors differ), $v^i := (\mathbf{T}[i][j_1], \ldots, \mathbf{T}[i][j_k])$, a_ℓ is the answer to $q(j_\ell)$, and $b_\ell := \begin{cases} 1 & a_\ell = \text{yes} \\ 0 & a_\ell = \text{no} \end{cases}$. Intuitively, since the adversary lies, the guesser can only "approximate" the chosen character, i.e., find a vector that differs from that of the chosen character by at most d bits. The uniqueness of i ensures that the guesser can determine who is the chosen character based on the approximate vector she gets.

3 Playing with Honest (but Sometimes Lazy) Players

We start by considering settings in which the chooser is honest, i.e., always answers correctly. The following result is based on the very basic strategy of binary search.

▶ **Theorem 4.** *For every n-character game, there is a $\lceil \log n \rceil$-question strategy. In addition, there is no k-question strategy with $k < \lceil \log n \rceil$.*

Proof. The theorem is achieved by a simple binary search algorithm, defined recursively. Given a set I_ℓ with $|I_\ell| > 1$, we choose $A \subseteq I_\ell$ with $|A| = \lfloor |I_\ell|/2 \rfloor$, and use Assumption 1 to find a trait j with $A = \{i \mid \mathbf{T}[i][j] = 1\}$. We ask $q_{\ell+1} = q(j)$ to get an answer $a_{\ell+1}$, and set $I_{\ell+1} = \{i \mid \mathbf{T}[i][j] = a_{\ell+1}\}$.

It is immediate to see that the algorithm defines a successful run $q_1, a_1, I_1, \ldots, q_k, a_k, I_k$ with $|I_k| = 1$. Indeed, using the inequality $|I_{\ell+1}| \leq \lceil |I_\ell|/2 \rceil \leq (|I_\ell|+1)/2$, a simple induction, and the fact that $|I_0| = n$, we get $|I_k| \leq \frac{n+2^k-1}{2^k}$. For $k = \lceil \log n \rceil \geq \log n$, this implies $|I_k| \leq 2 - 1/n$ and the integrality of $|I_k|$ guarantees $|I_k| = 1$ as claimed.

For the second part of the theorem, let us assume that there is a k-question strategy with questions $q_1, \ldots, q_k$. Each successful run that is consistent with this strategy is of the form $I_0, q_1, a_1, I_1, \ldots, q_k, a_k, I_k$ where for every ℓ, $a_\ell \in \{\text{yes}, \text{no}\}$ and $I_{\ell+1} \subseteq I_\ell$. Note that $I_{\ell+1}$ is uniquely determined by $a_1, \ldots, a_k$ and thus the number of possible I_ks is bounded by the number of possible vectors $a_1, \ldots, a_k$. Hence, there are at most 2^k successful runs that are consistent with this strategy. If $k < \lceil \log n \rceil$ then $2^k < n$ which implies that $q_1, \ldots, q_k$ cannot be a k-question strategy (since there are n characters). ◀

Notice that this strategy is adaptive, that is, at each round the guesser chooses the next question based on the outcome of the previous one. Is it possible to find a strategy that is not adaptive, that is, a strategy against an impatient adversary? We answer this in the affirmative.

▶ **Theorem 5.** *For every n-character game, there is a $\lceil \log n \rceil$-question predetermined-strategy.*

Proof. For $1 \leq \ell \leq \lceil \log n \rceil$, let $A_\ell = \{i \in [n] \mid i[\ell] = 1\}$, where $i[\ell]$ is the bit in location ℓ of the binary expansion of i (whose first bit is $i[1]$). For each ℓ in the range, we set q_ℓ to be the trait separating A_ℓ. (It exists due to Assumption 1.)

To prove this strategy always assures termination in $\lceil \log n \rceil$ questions, we consider the run $q_1, a_1, I_1, \ldots, q_{\lceil \log n \rceil}, a_{\lceil \log n \rceil}, I_{\lceil \log n \rceil}$ induced by the aforementioned questions. The choice of questions guarantees $I_\ell = \{i \mid i[1] = a_1, \ldots, i[\ell] = a_\ell\}$. Therefore, $I_{\lceil \log n \rceil}$ contains solely the element i with binary encoding $a_{\lceil \log n \rceil}\, a_{\lceil \log n \rceil - 1} \cdots a_2\, a_1$. ◀

This "static" approach can be useful when playing with a *Not-At-Time-ANswering* (NATAN) player that answers the questions altogether. Therefore, we can conclude the following.

▶ **Corollary 6.** *The classical Guess Who game has a 5-question predetermined-strategy, i.e., a strategy that works even against NATAN.*

4 Playing with Clueless Players

In this section, we consider *clueless* players, that do not always know the answer to a question. A clueless player might answer, in addition to "yes" and "no", the answer "I do not know" ("don't know", for brevity). One way to circumvent this answer is, whenever the player answers "don't know" on some trait, to ask a question on a different trait that has exactly

the same answers on all characters. However, this requires the assumption that such a trait always exists, and in addition, it does not work against an adversary that is both clueless and impatient. Instead, we suggest a different solution, using *erasure codes*.

▶ **Definition 7** ([20]). *An* erasure code *that can handle d erasures is a collection of binary vectors, namely* code-words*, such that there is a recovery algorithm that gets as an input each of these vectors with at most d locations replaced by "?", and returns the original vector.*

We restrict our attention to codes where all the code-words have equal length. A simple erasure code with $d = 1$ can be achieved, for example, by adding a parity bit – the recovery algorithm in this case will replace the "?" with a value complementing the number of 1-s in the vector to be even. A code that can handle more erasures can be achieved, e.g., by taking a collection of low-degree polynomials, and producing a code-word from each polynomial p by setting the i-th place of the code-word to $p(i)$.

Before presenting the result, let us set its formal setup.

▶ **Theorem 8.** *If there is an erasure code composed of at least n binary vectors of length k, that can handle d erasures, then for every n-character game there is a k-question predetermined-strategy against an adversary that answers "don't know" at most d times.*

Proof. Assign each character i with a unique vector v_i from the code. For $1 \leq \ell \leq k$, let $A_\ell = \{i \mid v_i[\ell] = 1\}$ be all the characters with the ℓ-th entry of their vector equal to 1. Let j_ℓ be the trait that separates A_ℓ from $\bar{A}_\ell$.

We use the question sequence $q_1 := q(j_1), \ldots, q_k := q(j_k)$, which defines a run $q_1, a_1, I_1, \ldots, q_k, a_k, I_k$. From the sequence $(a_1, \ldots, a_k)$ of answers, which contains at most d answers of "?", we find the unique way to complement the vector into a code-word $v' = (a'_1, \ldots, a'_k)$. The character i with $v_i = v'$ is then the desired character.

For correctness, consider the true character i^* chosen, and the sequence $(a^*_1, \ldots, a^*_k)$ of the true answers to the questions, where a^*_ℓ is the answer on $q(j_\ell)$. Note that the choice of questions guarantees that the following are equivalent: $a^*_\ell = 1$; $i^* \in A_\ell$; and $v_{i^*}[\ell] = 1$. This concludes in $v_{i^*}[\ell] = a^*_\ell$, from which we have $v_{i^*} = (a^*_1, \ldots, a^*_k)$. By the assumption of at most d answers are missing in the answer vector $(a_1, \ldots, a_k)$, and the properties of the erasure code ensure that the completion of this vector is unique, and is v_{i^*}. Hence, the algorithm returns i^*, as desired. ◀

One application of the above theorem is when the adversary may answer "don't know" at most once. For this, we use simple parity check bit. That is, by concatenating a parity bit to every $\lceil \log n \rceil$-bit binary vector, we get the following code.

▶ **Lemma 9.** *For any positive integer n, there is an erasure code of n vectors of length $\lceil \log n \rceil + 1$ allowing to overcome a single deletion.*

Recall that ARI is a player that might answer "don't know" at most once. By choosing $n = 24$ in the lemma and applying Theorem 8, we get the following corollary.

▶ **Corollary 10.** *The classical game of Guess Who has a 6-question predetermined-strategy against ARI.*

5 Playing with Liars

While in the previous section, the clueless player was honest (i.e., provided only correct answers), we now turn our attention to a player that might give incorrect answers. These are sometimes called Byzantine or malicious players, but we will give the kids the benefit of the doubt and say they are merely "mistaken".

A key tool in this section will be *error correcting codes*.

▶ **Definition 11.** *An* error correcting code *of distance d is a collection of equal-length binary vectors, such that the Hamming distance between every two is at least d.*

We will refer to such a code a "distance-d code". Note that, given such a code and a vector v, if there exists a code word c with Hamming distance at most $\lfloor \frac{d-1}{2} \rfloor$ from v, then c is the unique code-word with this property. The common use of these codes is thus by "correcting" each such vector v to the unique nearest code-word c; hence, we say that a distance-d code can *fix* at most $\lfloor \frac{d-1}{2} \rfloor$ errors.

▶ **Theorem 12.** *If there is a distance-d error correcting code composed of at least n binary vectors of length k, then for every n-character game there is a k-question predetermined-strategy against an adversary that performs at most $\lfloor \frac{d-1}{2} \rfloor$ mistakes.*

Proof. Assign each character i with a unique vector v_i from the code. For $1 \leq \ell \leq k$, let $A_\ell = \{i \mid v_i[\ell] = 1\}$ be all the characters with the ℓ-th entry of their vector equal to 1. Let j_ℓ be the trait that separates A_ℓ.

We use the question sequence $q_1 = q(j_1), \ldots, q_k = q(j_k)$, which defines a run $I_0, q_1, a_1, I_1, \ldots, q_k, a_k, I_k$. From the sequence $(a_1, \ldots, a_k)$ of answers, we find the unique closest code-word $v' = (a'_1, \ldots, a'_k)$, which differs from the sequence of answers by at most $\lfloor \frac{d-1}{2} \rfloor$ answers. The character i with $v_i = v'$ is then the desired character.

For correctness, consider the true character i^* chosen, and the sequence $(a^*_1, \ldots, a^*_k)$ of the true answers to the questions, where a^*_ℓ is the answer on $q(j_\ell)$. Note that the choice of questions guarantees that the following are equivalent: $a^*_\ell = 1$; $i^* \in A_\ell$; and $v_{i^*}[\ell] = 1$. This concludes in $v_{i^*}[\ell] = a^*_\ell$, from which we have $v_{i^*} = (a^*_1, \ldots, a^*_k)$. By the assumption of at most $\lfloor \frac{d-1}{2} \rfloor$ wrong answers, we know that the answer vector $(a_1, \ldots, a_k)$ differs from v_{i^*} in at most $\lfloor \frac{d-1}{2} \rfloor$ positions, which implies that the unique vector closest to the answer vector is indeed v_{i^*}, and the algorithm returns i^* as desired. ◀

One application of the above theorem is for the VAL player, that might make at most a single mistake. For this, we use a code guaranteed to exist by the Gilbert–Varshamov bound (see, e.g. [17]).

▶ **Lemma 13.** *For positive integers k, N satisfying $2^N < \frac{2^k}{k}$, there is a distance-3 code composed of at least 2^N binary vectors of length k.*

By choosing $k = 9$, $N = 5$ above, we get a code that implies, using Theorem 12, the following corollary.

▶ **Corollary 14.** *The classical game of Guess Who has a 9-question predetermined strategy against VAL.*

6 Concluding Remarks

As mentioned, Guess Who was invented as a simple version of the color-guessing Mastermind game. In Mastermind, the questions and answers are more involved than the binary questions asked in Guess Who, and thus, playing it with players that do not strictly follow the game protocol is an interesting direction we leave for future research.

Similar question holds for the trendy Wordle game, where the goal is not to guess a character but a word. Among the gallery of variations of Wordle that can be found online, one can consider a variant that does not always correctly mark the guessed letters, or where the user does not get feedback for the guesses until the end. These will probably not be the most fun-to-play version of the game, but poses interesting theoretical questions, such as error correction in natural language.

The commercial version of Guess Who was a target for criticism for its gallery of characters, where most characters are white males, and the exceptions (usually 5 out of 24) are women or people of color. One can consider a more inclusive version of the game, not only by splitting the traits more equally, but also by allowing non-binary answers to some questions. This could be done, e.g., by allowing questions such as "what is the color of the character's hair". A trinary version of the game (three hair colors, no-glasses/monocle/glasses, etc.) is yet to be presented, and the corresponding algorithmic questions are yet to be studied.

References

1 Aaron Berger, Christopher Chute, and Matthew Stone. Query complexity of mastermind variants. *Discret. Math.*, 341(3):665–671, 2018.

2 Lotte Berghman, Dries R. Goossens, and Roel Leus. Efficient solutions for mastermind using genetic algorithms. *Comput. Oper. Res.*, 36:1880–1885, 2009.

3 Lucas Boczkowski, Uriel Feige, Amos Korman, and Yoav Rodeh. Navigating in trees with permanently noisy advice. *ACM Trans. Algorithms*, 17(2):15:1–15:27, 2021.

4 Marco Console, Paolo Guagliardo, and Leonid Libkin. On querying incomplete information in databases under bag semantics. In *IJCAI*, volume 17, pages 993–999, 2017.

5 Carlos Cotta, Juan Julián Merelo Guervós, Antonio Mora García, and Thomas Philip Runarsson. Entropy-driven evolutionary approaches to the mastermind problem. In *PPSN*, 2010.

6 Yuval Dagan, Yuval Filmus, Daniel Kane, and Shay Moran. The entropy of lies: Playing twenty questions with a liar. In *ITCS*, volume 185 of *LIPIcs*, pages 1:1–1:16. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

7 Claire David, Leonid Libkin, and Filip Murlak. Certain answers for xml queries. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 191–202, 2010.

8 Dariusz Dereniowski, Daniel Graf, Stefan Tiegel, and Przemyslaw Uzna'nski. A framework for searching in graphs in the presence of errors. In *SOSA*, 2019.

9 Julien Gagneur, Markus C. Elze, and Achim Tresch. Selective phenotyping, entropy reduction, and the mastermind game. *BMC Bioinformatics*, 12:406–406, 2011.

10 Juan Julián Merelo Guervós, Pedro A. Castillo, Antonio Mora García, and Anna I. Esparcia-Alcázar. Improving evolutionary solutions to the game of mastermind using an entropy-based scoring method. In *GECCO '13*, 2013.

11 Juan Julián Merelo Guervós, Antonio Mora García, Pedro A. Castillo, Carlos Cotta, and Mario García Valdez. A search for scalable evolutionary solutions to the game of mastermind. *2013 IEEE Congress on Evolutionary Computation*, pages 2298–2305, 2013.

12 Nicolas Hanusse, David Ilcinkas, Adrian Kosowski, and Nicolas Nisse. Locating a target with an agent guided by unreliable local advice: how to beat the random walk when you have a clock? *Proceedings of the 29th ACM SIGACT-SIGOPS symposium on Principles of distributed computing*, 2010.

13 Nicolas Hanusse, Evangelos Kranakis, and Danny Krizanc. Searching with mobile agents in networks with liars. In *Euro-Par*, 2000.

14 Tomasz Imieliński and Witold Lipski Jr. Incomplete information in relational databases. In *Readings in Artificial Intelligence and Databases*, pages 342–360. Elsevier, 1989.

15 Donald Ervin Knuth. The computer as master mind. *Journal of Recreational Mathematics*, 9:1–6, 1977.

16 Witold Lipski Jr. On databases with incomplete information. *Journal of the ACM*, 28(1):41–70, 1981.

17 Florence Jessie MacWilliams and Neil James Alexander Sloane. *The theory of error correcting codes*, volume 16. Elsevier, 1977.

18 Mihai Nica. Optimal strategy in "guess who?": Beyond binary search. *Probability in the Engineering and Informational Sciences*, 30(4):576–592, 2016. doi:10.1017/S026996481600022X.

19 Riccardo Rosati. On the decidability and finite controllability of query processing in databases with incomplete information. In *Proceedings of the twenty-fifth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 356–365, 2006.

20 Ron Roth. *Introduction to Coding Theory*. Cambridge University Press, 2006. doi:10.1017/CBO9780511808968.

21 Geoffroy Ville. An optimal mastermind (4,7) strategy and more results in the expected case. *ArXiv*, abs/1305.1010, 2013.

22 Jef Wijsen. On the first-order expressibility of computing certain answers to conjunctive queries over uncertain databases. In *Proceedings of the twenty-ninth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 179–190, 2010.