Query Complexity of Mastermind Variants

Aaron Berger Christopher Chute Matthew Stone

December 26, 2015

Abstract

We study variants of Mastermind, a popular board game in which the objective is sequence reconstruction. In this two-player game, the codemaker constructs a hidden sequence $H = (h_1, h_2, \ldots, h_n)$ of colors selected from an alphabet $\mathcal{A} = \{1, 2, \ldots, k\}$ (*i.e.,* $h_i \in \mathcal{A}$ for all $i \in \{1, 2, \ldots, n\}$). The game then proceeds in turns, each of which consists of two parts: in turn t, the codebreaker first submits a query sequence $Q_t = (q_1, q_2, \ldots, q_n)$ with $q_i \in \mathcal{A}$ for all i, and second receives feedback $\Delta(Q_t, H)$, where Δ is a function of distance between two n-sequences. The game terminates when $Q_t = H$, and the codebreaker seeks to end the game in as few turns as possible. Throughout we let $f(n, k)$ denote the smallest integer such that the codebreaker can determine any H in $f(n, k)$ turns.[1] We prove three main results: First, we show that given a set S_t of sequences which is known to contain H, there always exists a query whose feedback implicitly produces a smaller set S_{t+1} such that $H \in S_{t+1}$ and $|S_{t+1}| \leq (1 - 1/(nk))|S|$. Second, when H is known to be a permutation of the alphabet $\{1, 2, \ldots, n\}$, we prove that $f(n, n) \geq n - \log\log(n)$ for all sufficiently large n. Third, when feedback is not received until all queries have been submitted, we show that there exists a constant $c > 0$ such that $f(n, k) \geq c \cdot n \log(k)$.

1 Introduction

Variants of Mastermind are a family of two-player games centered around reconstruction of a hidden sequence. In all variants of the game, one player is given the role of "codemaker," and the other is denoted the "codebreaker." The codemaker begins the game by constructing a hidden sequence $H = (h_1, h_2, \ldots, h_n)$ where each component is selected from an alphabet $\mathcal{A} = \{1, 2, \ldots, k\}$ of k colors (that is, $h_i \in \mathcal{A}$ for all $i \in \{1, 2, \ldots, n\}$). The goal of the codebreaker is to uniquely determine the hidden sequence H through a series of queries, which are submissions of vectors of the form $Q_t = (q_1, q_2, \ldots, q_n)$. The codebreaker always seeks to determine H with as few queries as possible, however the nature of these queries, the feedback received after a query, and the restrictions on H differ between variants.

The variants which we will study are differentiated by settings of the tuple (n, k, Δ, R, A). These parameters are defined as follows:

(i) (n) *Length of Sequence.* The parameter n denotes the length of the hidden sequence H created by the codemaker, hence $H = (h_1, h_2, \ldots, h_n)$. The codebreaker is also required to submit query vectors of length n, so the t^{th} query vector takes the form $Q_t = (q_1, q_2, \ldots, q_n)$.

(ii) (k) *Size of Alphabet.* This parameter determines the number of possible values for components of H and Q_t. Each game is accompanied by an alphabet $\mathcal{A} = \{1, 2, \ldots, k\}$ from which the components of H and Q_t are selected. That is, $h_i \in \{1, 2, \ldots, k\}$ and $q_i \in \{1, 2, \ldots, k\}$.

(iii) (Δ) *Distance Function.* On the t^{th} turn of a game, the codebreaker submits a query sequence $Q_t = (q_1, q_2, \ldots, q_n)$. The codemaker then gives feedback $\Delta(Q_t, H)$, which is roughly a measure of distance between Q_t and H. The information yielded by $\Delta(Q_t, H)$ may be used to guide the choice of Q_{t+1}, the next sequence to be guessed. Hence the choice of distance function affects the codebreaker's ability to make an informed query on the following turn. We study the following distance functions:

  a. *"Black-peg and white-peg."* Let Q_t and H be as above. The black-peg and white-peg distance function is defined by $\Delta(Q_t, H) = (b(Q_t, H), w(Q_t, H))$ where

$$b(Q_t, H) = \left|\{i \in \mathbb{Z} \mid q_i = h_i,\ 1 \le i \le n\}\right|, \tag{1}$$

  and

$$w(Q_t, H) = \max_{\sigma \in S_{Q_t}} \; b(\sigma(Q_t), H) - b(Q_t, H),$$

  We note that this is the distance function used in the original game of Mastermind.

  b. *"Black-peg-only."* When Δ is the black-peg-only distance function, it is defined by $\Delta(Q_t, H) = b(Q_t, H)$, where b is defined as in equation (1).

(iv) (R) *Repetition.* The parameter R is a Boolean restriction on the components of H. If R is true, we say that the variant game is *with repeats* or that repeats are allowed. In this case, the hidden vector H may have repeated colors, that is, we allow $h_i = h_j$ for any $i, j \in \{1, 2, \ldots, n\}$. When R is false, we say that the variant is *no repeats*, and we require $h_i \neq h_j$ when $i \neq j$. In particular, when R is false and $k = n$, we have that H must be a permutation of $(1, 2, \ldots, n)$, and we refer to this variant as the *Permutation Game.*

(v) (A) *Adaptiveness.* The Boolean parameter A determines whether the codebreaker receives feedback after each query. If A is true, we say that the game is *adaptive.* In this case the game consists of two-part turns: on the t^{th} turn, the codebreaker first submits a query sequence Q_t, and then receives feedback $\Delta(Q_t, H)$. The codebreaker may use the feedback to inform the query Q_{t+1} made in turn $t + 1$, and the game ends in turn s if and only if $Q_s = H$.

  When A is false, we say that the game is *non-adaptive.* In this case the codebreaker submits m queries $Q_1, Q_2, \ldots, Q_m$ all at once (where the codebreaker chooses m). The codemaker then reports a feedback vector of the form $(\Delta(Q_1, H), \Delta(Q_2, H), \ldots, \Delta(Q_m, H))$, after which the codebreaker must submit the final query $\overline{Q}$. The codebreaker wins if and only if $\overline{Q} = H$.

Throughout we define $f(n, k, \Delta, R, A)$ to be the smallest integer such that the codebreaker can determine any hidden sequence H in $f(n, k, \Delta, R, A)$ queries during a game with the corresponding assignment of n, k, Δ, R, and A. We will denote true and false by T and F, respectively, for assignment of Boolean variables. For example, Donald Knuth's result that the original game of Mastermind (four positions, six colors, black-peg and white-peg, with repeats, and adaptive) can always be determined after four turns is equivalently stated as $f(4, 6, \Delta = (b, w), R = T, A = T) \le 4$ (in fact, this bound holds with equality). We will write simply $f(n, k)$ when the context is clear.

We prove three main results, each of which builds on the previous work related to a given Mastermind variant.

Theorem 1. Let $A = T$, let Δ be the black-peg-only distance function, and let n, k, and R be fixed. Given a set S_t of sequences which is known to contain H, there exists a query whose feedback produces a smaller set S_{t+1} such that $H \in S_{t+1}$ and $|S_{t+1}| \le (1 - 1/(nk))|S|$.

This theorem facilitates analysis of a common variant of Mastermind, in which adaptive queries are made with feedback given by the black-peg-only distance function. To demonstrate the theorem's usefulness, suppose that the codebreaker is in turn t, hence has made queries $Q_1, Q_2 \ldots, Q_{t-1}$ and received feedback instances $\Delta(Q_1, H), \Delta(Q_2, H) \ldots, \Delta(Q_{t-1}, H)$. Given this set of feedback, the codebreaker can eliminate a number of possibilities for H. For example, suppose that a sequence $\tilde{H} = (\tilde{h}_1, \tilde{h}_2 \ldots, \tilde{h}_n)$ satisfies $\Delta(\tilde{H}, H) \neq \Delta(Q_s, H)$ for some $s \in \{1, 2, \ldots, t-1\}$. Then clearly $\tilde{H} \neq H$.

Using this process, the codebreaker may construct a set S_t of possibilities for H in turn t after $t-1$ query-feedback interactions have occurred. That is, S_t contains sequences of length n with components from the alphabet $\mathcal{A}$ such that no member of S_t has been eliminated by the responses $\Delta(Q_1, H), \Delta(Q_2, H), \ldots, \Delta(Q_{t-1}, H)$. Theorem 1 then states that given S_t, there always exists a query Q_t such that the response $\Delta(Q_t, H)$ will eliminate a fraction $1/(nk)$ of the members of S_t as possibilities for H. By Theorem 1, $|S_{t+1}| \leq (1 - 1/(nk))|S_t|$, which guarantees a quantifiable amount of progress at each turn of an adaptive, black-peg-only variant of mastermind.

Our second main result concerns the Permutation Game, in which $n = k$ and $R = F$, thus restricting the hidden sequence H to be a permutation of the alphabet $\mathcal{A}$.

Theorem 2. Consider the Permutation Game defined by $n = k$ and $R = F$. Let $\Delta = b$ (black-peg-only distance function) and let A be fixed. Then for all sufficiently large n, we have

$$f(n, k = n, \Delta = b, R = F, A) \geq n - \log\log(n).$$

Explicit algorithms that take $O(n \log n)$ turns to solve this variant were developed by Ko and Teng[2], and Ouali and Sutherland[3]. Ko and Teng approach the problem with an algorithm akin to binary search. The algorithm queries a sequence Q_t, swaps two components, and queries again, doing so repeatedly until a previously unknown component of H is determined by the values of the distance function across these repeated queries. Ouali and Sutherland improve this algorithm primarily by altering the search routine after a single component of H has been identified. In this way, Ouali and Sutherland achieve an average factor of 2 reduction in the number of queries needed to identify H. In our notation, these results state that there exists a constant $c > 0$ such that $f(n, k = n, \Delta = b, R = F, A = T) \leq c \cdot n \log(n)$.

Via a basic information-theoretic argument, one can show that the Permutation Game also satisfies $f(n, n) \geq n - n/\log(n) + c$ for some constant $c > 0$. We improve this lower bound to $f(n, n) \geq n - \log(n)$ for small n, and $f(n, n) \geq n - \log\log(n)$ for sufficiently large n. To our knowledge, this constitutes the first improvement over the trivial information-theoretic lower bound for the Permutation Game variant of Mastermind.

Our third result relates to non-adaptive variants of Mastermind, which correspond to $A = F$ in our notation.

Theorem 3. Consider non-adaptive Mastermind, defined by $A = F$. Let Δ be the black-peg-only distance function, and let n, k, and R be fixed. Then there exists a constant $c > 0$ such that

$$f(n, k, \Delta = b, R, A = F) \geq c \cdot n \log(k).$$

To our knowledge, there have been no previous attempts to provide a lower bound for $f(n, k)$ in non-adaptive variants of Mastermind. With regard to an upper bound, we have the following result due to Vaclav Chvatal:[1] *For every positive ϵ there is an $n(\epsilon)$ with the following property: if $n > n(\epsilon)$ and if $k < n^{1-\epsilon}$ then*

$$f(n, k, \Delta = b, R, A = F) \leq (2 + \epsilon) n \frac{1 + 2\log(k)}{\log(n) - \log(k)}.$$

Chvatal's proof of the above result relies on a probabilistic argument, showing that there always exists a set of $(2+\epsilon)(n)(1+2\log(k))/(\log(n)-\log(k))$ queries such that the responses (*i.e.,* values of $\Delta(Q_t, H)$) will differentiate between each possible hidden vector H.

2 Adaptive Variants of Mastermind

2.1 The Permutation Game

We begin by performing an analysis of the Permutation Game, defined by $n = k$ and $R = F$. These parameters together imply $H = \sigma(\mathcal{A})$ for some permutation $\sigma \in S_k$ (recalling that $\mathcal{A} = \{1, 2, \ldots, k\}$). As stated in the introduction, an upper bound $f(n, n) \leq c \cdot n \log(n)$ was proved in 1986 by Ko and Teng, and again in 2013 for a smaller constant $c > 0$ by Ouali and Sutherland. Here we establish lower bounds for $f(n, n) = f(n, k = n, \Delta = b, R = F, A = T)$ concerning the Permutation Game.

2.1.1 Trivial Lower Bound

We begin by establishing the trivial information-theoretic result that

$$f(n, n) \geq \log_n(n!) = n - \frac{n}{\ln(n)} + O(1).$$

Proof. Consider an instance of the Permutation Game with the black-peg-only distance function Δ. To uniquely identify H after m turns, the query response vector $(\Delta(Q_1, H), \Delta(Q_2, H), \ldots, \Delta(Q_m, H))$ must distinguish between all $n!$ permutations of the alphabet $\mathcal{A}$. In the t^{th} turn, the codebreaker submits a query sequence Q_t, which has n possible associated responses given by $\Delta(Q_t, H) \in \{0, 1, \ldots, n-2, n\}$. We note that it is not possible to get $\Delta(Q_t, H) = n - 1$ in the Permutation Game. Therefore there are n^m possible query response vectors of length m, hence to identify H in m queries requires that $n^m \geq n!$. This implies $m \geq \log_n(n!)$, and the result follows immediately. □

2.1.2 Buckets

To improve upon the trivial lower bound, we first introduce terminology with which to analyze the distribution of the possible solutions after feedback $\Delta(Q_t, H)$ from a given query Q_t. Let Δ be the black-peg-only distance function. Given a query vector Q_t, the codebreaker may partition the set S_t of remaining solutions into subsets which we will call *buckets*; one bucket for each possible response given by the codemaker. We now formalize the notion of a bucket.

Consider the t^{th} turn in an instance of the Permutation Game. As described in the introduction, the codebreaker may use the responses $\Delta(Q_1, H), \Delta(Q_2, H), \ldots, \Delta(Q_{t-1}, H)$ to construct a minimal set S_t of sequences such that $H \in S_t$. Suppose the codebreaker submits the query $Q_t = (q_1, q_2, \ldots, q_n)$. Then we define the *bucket* $B_t(s)$ *implied by response* $\Delta(Q_t, H) = s$ to be

$$B_t(s) = \{X \in S_t \mid \Delta(Q_t, X) = s\}.$$

Such a construct is useful in that if $\Delta(Q_t, H) = s$, the codebreaker may deduce that if a sequence $X \in S_t$ satisfies $\Delta(Q_t, X) \neq s$, then $X \neq H$. Therefore the codebreaker may construct the set S_{t+1} of possible solutions after the t^{th} turn is complete by setting $S_{t+1} = B_t(s)$.

We now compute the size of a bucket $B_1(s)$ produced in the first turn of an instance of the Permutation Game. First we choose the s colors that are fixed points with respect to the query sequence Q_1. We then permute the remaining $n-s$ colors without any fixed points, giving $D(n-s)$, the derangements on an $(n-s)$-element vector. Hence we have

$$|B_1(s)| = \binom{n}{s} D(n-s),$$

for the bucket implied by response $\Delta(Q_1, H) = s$. Clearly the buckets $B_1(1), B_1(2), \ldots, B_1(n)$ partition the set S_1 of $n!$ possible hidden vectors.

The above computation tells us the *initial* size of each bucket, which is independent of the initial query Q_1. Now consider the evolution of bucket sizes, *i.e.,* the sequence $|B_1(s)|, |B_2(s)|, |B_3(s)|, \ldots$ over the course of an instance of the Permutation Game. We claim that this is a non-increasing sequence. Since $|B_1(s)|$ is independent of Q_1, assume that $Q_1 = Q_t$. Then the query Q_t, together with response $\Delta(Q_t, H)$, produces a bucket $B_t(s)$ defined by $B_1(s) \cap S_t$. Since $|S_1|, |S_2|, |S_3|, \ldots$ is clearly a non-increasing sequence, it must be the case that $|B_t(s)| = |B_1(s) \cap S_t| \geq |B_1(s) \cap S_{t+1}| = |B_{t+1}(s)|$. Hence bucket sizes are bounded above by their initial size, *i.e.,*

$$|B_t(s)| \leq |B_1(s)| = \binom{n}{s} D(n-s).$$

2.1.3 Improved Lower Bound for the Permutation Game

We now employ the notion of a bucket to improve the trivial lower bound for $f(n,n)$ in the context of the Permutation Game.

Theorem 2. Consider the Permutation Game defined by $n = k$ and $R = F$. Let $\Delta = b$ (black-peg-only distance function) and let A be fixed. Then for all sufficiently large n, we have

$$f(n, k = n, \Delta = b, R = F, A) \geq n - \log\log(n).$$

Proof. Let n be fixed. We give a proof which analyzes the evolution of $L(t)$, which we define to be a lower bound on $|S_t|$. We recall that S_t is the minimal set of sequences guaranteed to contain H given the responses $\Delta(Q_1, H), \Delta(Q_2, H), \ldots, \Delta(Q_{t-1}, H)$. We compute this lower bound by using a hypothetical set of queries $\{Q_1, Q_2, \ldots, Q_{t-1}\}$, constructed such that Q_i maximizes the difference $|S_i| - |S_{i-1}|$. That is, each Q_i guarantees to "rule out" the maximal number possibilities for H in the i^{th} turn. The least rapidly decreasing sequence $|S_1|, |S_2|, |S_3|, \ldots$ for this set of queries will converge to $|S_i| = |S_{i+1}| = 1$, and the turn in which this convergence is initially reached will provide a lower bound for $f(n,n)$. Our approach is to find a number r such that that $t \geq r$ implies $L(t) = 1$. We proceed by induction on t.

Consider the t^{th} turn in an instance of the Permutation Game. In the terminology of buckets, the codebreaker constructs the set S_t in turn t by setting $S_t = B_{t-1}(\Delta(Q_{t-1}, H))$. To choose an "optimal" query sequence Q_t (in the sense described above, which maximizes $|S_{t+1}| - |S_t|$), the codebreaker therefore must choose a sequence Q_t which minimizes

$$\max_{s \in \{1,2,\ldots,n\}} |B_{t+1}(s)|.$$

Equivalently, the codebreaker chooses Q_t such that the size of the largest bucket is as small as possible. By the discussion in Section (2.1.2), we know that $|B_t(s)| \leq \binom{n}{s} D(n-s)$. The optimal query Q_t will partition S_t into buckets as evenly as possible, and some buckets will have size equal to their upper bound, while others will be partially filled. This best-possible partition will partition S_t into two sets X and Y of buckets, such that $A \in X$ implies bucket A is incompletely filled, $B \in Y$ implies bucket B

is completely filled, and $A \in X$, $B \in Y$ implies that the upper bound on $|A|$ is greater than the upper bound on $|B|$. Let $x = |X|$, so that x is the number of incompletely filled buckets.

We now give a recurrence which bounds $L(t)$. At the beginning of the t-th turn, we have at least $L(t-1)$ possible solutions remaining, so $|S_t| \geq L(t-1)$. Suppose that the codebreaker has selected the optimal query vector Q_t for submission in turn t. We then have

$$\begin{aligned} L(t-1) &\leq \sum_{i=0}^{n} |B_i(t)| \\ &= \sum_{i=0}^{x-1} |B_i(t)| + \sum_{i=x}^{n} |B_i(t)| \\ &\leq \sum_{i=0}^{x-1} L(t) + \sum_{i=x}^{n} \binom{n}{i} D(n-i) \quad \text{for optimal } x \\ &= x \cdot L(t) + \sum_{i=x}^{n} \binom{n}{i} D(n-i). \end{aligned} \tag{2}$$

Once we know $L(t)$, picking x too large results in some of the smaller buckets being sub-optimally underfilled. Picking x too small results in some of the larger buckets being overfilled. So adjustment of x about its optimal value always increases the value of equation (2). Thus we have

$$L(t-1) \leq x \cdot L(t) + \sum_{i=x}^{n} \binom{n}{i} D(n-i) \quad \forall x$$

Where t is the number of turns taken, $L(t)$ is a lower bound on $|S_{t+1}|$ (the number of possible solutions remaining after the t-th turn), and D is the derangements function.

We now prove the following lemma.

Lemma 1.

$$\sum_{i=x}^{n} \binom{n}{i} D(n-i) \leq \frac{n!}{x!}$$

Proof. A combinatorial argument can be made as follows: The LHS denotes the number of permutations of an n-element vector which have at least x fixed points. Suppose we choose x fixed points and simply permute the rest of the vector. This gives $\binom{n}{x}(n-x)! = \frac{n!}{x!}$ possible permutations. Clearly this includes all vectors with at least x fixed points (and over-counts by some margin), so the inequality holds.

One may also observe

$$\begin{aligned}
\sum_{i=x}^{n} \binom{n}{i} D(n-i) &= \frac{n!}{x!} \sum_{i=x}^{n} \frac{x!}{i!(n-i)!} D(n-i) \\
&\quad \text{we use the fact that } \frac{x!}{i!} \le \frac{1}{(i-x)!} \\
&\le \frac{n!}{x!} \sum_{i=x}^{n} \frac{1}{(i-x)!(n-i)!} D(n-i) \\
&\quad \text{let } k = i - x \\
&= \frac{n!}{x!} \sum_{k=0}^{n-x} \frac{1}{k!(n-x-k)!} D(n-x-k) \\
&= \frac{n!}{x!(n-x)!} \sum_{k=0}^{n-x} \binom{n-x}{k} D(n-x-k) \\
&= \frac{n!}{x!(n-x)!} (n-x)! \\
&= \frac{n!}{x!}.
\end{aligned}$$

□

Using Lemma 1, we have

$$L(t-1) \le x \cdot L(t) + \frac{n!}{x!},$$

which gives

$$L(t) \ge \frac{1}{x} \left(L(t-1) - \frac{n!}{x!} \right).$$

The following lemma will also be of use.

Lemma 2. We recall that n is fixed in defining $L(t)$. For any constant C_n, we have

$$\frac{L(t)}{n!} \ge \frac{C_n! - (H_{C_n+t} - H_{C_n})}{(C_n+t)!}$$

where $0 \le t \le n - C_n$ and $H_n = \sum_{i=1}^{n} \frac{1}{i}$ is the n^{th} harmonic number.

Proof. While this equation holds for all x, to achieve a desirable bound, we will let $x = t + C_n$ for any positive constant C_n. We have

Base Case. When $t = 0$, we have $S_n(0) = n!$ and

$$\frac{L(0)}{n!} \ge \frac{C_n! - (H_{C_n} - H_{C_n})}{C_n!} = 1$$

Inductive Step. Assume the induction hypothesis holds for $t-1$.

$$\frac{L(t)}{n!} \geq \frac{1}{x}\left(\frac{L(t-1)}{n!} - \frac{1}{x!}\right)$$

Let $x = C_n + t$.

$$\frac{L(t)}{n!} \geq \frac{1}{C_n+t}\left(\frac{L(t-1)}{n!} - \frac{1}{(C_n+t)!}\right)$$

Inductively,

$$\begin{aligned}
\frac{L(t)}{n!} &\geq \frac{1}{C_n+t}\left(\frac{C_n! - (H_{C_n+t} - H_{C_n})}{(C_n+t-1)!} - \frac{1}{(C_n+t)!}\right) \\
&\geq \left(\frac{C_n! - (H_{C_n+t-1} - H_{C_n}) - \frac{1}{C_n+t}}{(C_n+t)!}\right) \\
&\geq \left(\frac{C_n! - (H_{C_n+t} - H_{C_n})}{(C_n+t)!}\right)
\end{aligned}$$

Then

$$\begin{aligned}
L(n-C_n) &\geq n!\left(\frac{C_n! - (H_n - H_{C_n})}{n!}\right) \\
&\geq C_n! - (H_n - H_{C_n})
\end{aligned}$$

Thus the inequality holds for all n.

$\square$

We choose C_n by the following reasoning: we want to achieve a lower bound of $n - C_n$ turns. Thus, we want

$$C_n! - (H_n - H_{C_n}) > 1.$$

When $C_n = \log\log n$, we have $C_n! - (H_n - H_{C_n}) > 1$ for $n \geq e^{198}$, and when $C_n = \log n$, this is true for $n \geq 10$.

In conclusion, when n is sufficiently large, the minimum number of remaining possible solutions after $n - \log\log n$ guesses is at least

$$\begin{aligned}
L(n - \log\log n) &\geq (\log\log n)! - (H_n - H_{\log\log n}) \\
&> 1
\end{aligned}$$

Thus there is no strategy that can identify any hidden sequence in $n - \log\log(n)$ turns or fewer. Therefore, in the Permutation Game variant of Mastermind, we have $f(n,n) \geq n - \log\log(n)$ for all sufficiently large n. $\square$

References

[1] Chvátal, V.

[2] Ko, K.-I., and Teng, S.-C. On the number of queries necessary to identify a permutation. *Journal of Algorithms 7* (7 1986), 449–462.

[3] Ouali, M. E., and Sauerland, V. Improved approximation algorithm for the number of queries necessary to identify a permutation. *arXiv 1* (3 2013), 1–415.