Playing Mastermind With Many Colors

BENJAMIN DOERR, Max-Planck-Institut für Informatik
CAROLA DOERR, Max-Planck-Institut für Informatik and Université Paris Diderot
RETO SPÖHEL, Max-Planck-Institut für Informatik
HENNING THOMAS, ETH Zürich

42

We analyze the general version of the classic guessing game Mastermind with n positions and k colors. Since the case $k \leq n^{1-\varepsilon}$, $\varepsilon > 0$ a constant, is well understood, we concentrate on larger numbers of colors. For the most prominent case $k = n$, our results imply that Codebreaker can find the secret code with $O(n \log \log n)$ guesses. This bound is valid also when only black answer pegs are used. It improves the $O(n \log n)$ bound first proven by Chvátal. We also show that if both black and white answer pegs are used, then the $O(n \log \log n)$ bound holds for up to $n^2 \log \log n$ colors. These bounds are almost tight, as the known lower bound of $\Omega(n)$ shows. Unlike for $k \leq n^{1-\varepsilon}$, simply guessing at random until the secret code is determined is not sufficient. In fact, we show that an optimal nonadaptive strategy (deterministic or randomized) needs $\Theta(n \log n)$ guesses.

CCS Concepts: • **Theory of computation → Randomness, geometry and discrete structures**; **Random search heuristics;**

Additional Key Words and Phrases: Combinatorial games, Mastermind, query complexity, randomized algorithms

ACM Reference Format:
Benjamin Doerr, Carola Doerr, Reto Spöhel, and Henning Thomas. 2016. Playing Mastermind with many colors. J. ACM 63, 5, Article 42 (November 2016), 23 pages.
DOI: http://dx.doi.org/10.1145/2987372

1. INTRODUCTION

Mastermind (see Section 1.1 for the rules) and other guessing games like liar games [Pelc 2002; Spencer 1994] have attracted the attention of computer scientists not only because of their playful nature, but more importantly because of their relation to fundamental complexity and information-theoretic questions. In fact, Mastermind with two colors was first analyzed by Erdős and Rényi [1963] in 1963, several years before the release of Mastermind as a commercial board game.

Since then, intensive research by various scientific communities produced a plethora of results on various aspects of the Mastermind game (see also the literature review

C. Doerr is a recipient of the Google Europe Fellowship in Randomized Algorithms. This research was supported in part by this Google Fellowship. Her work was also partially supported by a Feodor Lynen Research Fellowship for Postdoctoral Researchers of the Alexander von Humboldt Foundation and by the Agence Nationale de la Recherche under project ANR-09-JCJC-0067-01.

Authors' addresses: B. Doerr, École Polytechnique, LIX–UMR 7161, 1 rue Honoré d'Estienne d'Orves, Bâtiment Alan Turing, Campus de l'École Polytechnique, 91120 Palaiseau, France; email: doerr@lix.polytechnique.fr; C. Doerr, Sorbonne Universités, UPMC University Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu 75005 Paris, France; email: Carola.Doerr@mpi-inf.mpg.de; R. Spöhel, Bern University of Applied Sciences, Engineering and Information Technology, 3400 Burgdorf, Switzerland; email: rspoehel@mpi-inf.mpg.de; H. Thomas, Institute of Theoretical Computer Science, ETH Zürich, 8092 Zürich, Switzerland; email: hthomas@inf.ethz.ch.

DOI: http://dx.doi.org/10.1145/2987372

Fig. 1. Typical round of Mastermind.

in Section 1.4). In a famous 1983 paper, Chvátal [1983] determined, precisely up to constant factors, the asymptotic number of queries needed on a board of size n for all numbers k of colors with $k \leq n^{1-\varepsilon}$, $\varepsilon > 0$ a constant. Interestingly, a very simple guessing strategy suffices, namely asking random guesses until the answers uniquely determine the secret code.

Surprisingly, for larger numbers of colors, no sharp bounds exist. In particular, for the natural case of n positions and $k = n$ colors, Chvátal's bounds $O(n \log n)$ and $\Omega(n)$ from 1983 are still the best known asymptotic results.

We almost close this gap open for roughly 30 years and prove that Codebreaker can solve the $k = n$ game using only $O(n \log\log n)$ guesses. This bound, as Chvátal's, even holds for black pegs–only Mastermind. When white answer pegs are used as well, we obtain a similar improvement from the previous best $O(n \log n)$ bound to $O(n \log\log n)$ for all $n \leq k \leq n^2 \log\log n$.

1.1. Mastermind

Mastermind is a two-player board game invented in the 1970s by the Israeli telecommunication expert Mordechai Meirowitz (Figure 1 shows a European version of the board). The first player, called *Codemaker* here, privately chooses a color combination of four pegs. Each peg can be chosen from a set of six colors. The goal of the second player, *Codebreaker*, is to identify this secret code. To do so, he guesses arbitrary length-4 color combinations. For each such guess, he receives information of how close his guess is to Codemaker's secret code. Codebreaker's aim is to use as few guesses as possible.

Besides the original four-position six-color Mastermind game, various versions with other numbers of positions or colors are commercially available. The scientific community, naturally, often regards a generalized version with n positions and k colors (according to Chvátal [1983], this was first suggested by Pierre Duchet). For a precise description of this game, let us denote by $[k]$ the set $\{1, \ldots, k\}$ of positive integers not exceeding k. At the start of the game, Codemaker chooses a secret code $z \in [k]^n$. In each round, Codebreaker guesses a string $x \in [k]^n$. Codemaker replies with the numbers $\mathrm{eq}(z, x) := |\{i \in [n] \mid z_i = x_i\}|$ of positions in which his and Codebreaker's string coincide, and with $\pi(z, x)$, the number of additional pegs having the right color but being in the wrong position. Formally, $\pi(z, x) := \max_{\rho \in S_n} |\{i \in [n] \mid z_i = x_{\rho(i)}\}| - \mathrm{eq}(z, x)$, where S_n denotes the set of all permutations of the set $[n]$. In the original game, $\mathrm{eq}(z, x)$ is indicated by black answer pegs, and $\pi(z, x)$ is indicated by white answer pegs. Based on this and all previous answers, Codebreaker may choose his next guess. He "wins" the game if his guess equals Codemaker's secret code.

We should note that often, and partially also in this work, a black pegs–only variant is studied, in which Codemaker reveals $\mathrm{eq}(z, x)$ but not $\pi(z, x)$. This is justified both by several applications (see Section 1.4) and by the insight that, particularly for small numbers of colors, the white answer pegs do not significantly improve Codebreaker's situation (see Section 3).

1.2. Previous Results

Mastermind has been studied intensively in the mathematics and computer science literature. For the original four-position six-color version, Knuth [1977] has given a deterministic strategy that wins the game in at most five guesses. He also showed that no deterministic strategy has a four-round guarantee.

The generalized n-position k-color version was investigated by Chvátal [1983]. He noted that a simple information-theoretic argument (attributed to Pierre Duchet) provides a lower bound of $\Omega(n \log k / \log n)$ for any $k = k(n)$.

Extending the result of Erdős and Rényi [1963] from $k = 2$ to larger numbers of colors, he then showed that for any fixed $\varepsilon > 0$, n sufficiently large and $k \le n^{1-\varepsilon}$, repeatedly asking random guesses until all but the secret code are excluded by the answers is an optimal Codebreaker strategy (up to constant factors). More specifically, using the probabilistic method and random guesses, he showed the existence of a deterministic nonadaptive strategy for Codebreaker—that is, a set of $(2+\varepsilon)n\frac{1+2\log k}{\log(n/k)}$ guesses such that the answers uniquely determine any secret code that Codemaker might have chosen (here and in the remainder, $\log n$ denotes the binary logarithm of n). These bounds hold even in the black pegs–only version of the game.

For larger values of k, the situation is less understood. Note that the information-theoretic lower bound is $\Omega(n)$ for any number $k = n^{\alpha}$, $\alpha > 0$ a constant, of colors. For k between n and n^2, Chvátal presented a deterministic adaptive strategy using $2n \log k + 4n$ guesses. For $k = n$, this strategy does not need white answer pegs. Chvátal's result has been improved subsequently. Chen et al. [1996] showed that for any $k \ge n$, $2n\lceil \log n \rceil + 2n + \lceil k/n \rceil + 2$ guesses suffice. Goodrich [2009b] proved an upper bound of $n\lceil \log k \rceil + \lceil (2 - 1/k)n \rceil + k$ for the number of guesses needed to win the Mastermind game with an arbitrary number k of colors and black answer pegs only. This was again improved by Jäger and Peczarski [2011], who showed an upper bound of $n\lceil \log n \rceil - n + k + 1$ for the case $k > n$ and $n\lceil \log k \rceil + k$ for the case $k \le n$. Note that for the case of $k = n$ colors and positions, all of these results give the same asymptotic bound of $O(n \log n)$.

1.3. Our Contribution

The preceding results show that Mastermind is well understood for $k \le n^{1-\varepsilon}$, where we know the correct number of queries apart from constant factors. In addition, a simple nonadaptive guessing strategy suffices to find the secret code, namely casting random guesses until the code is determined by the answers.

On the other hand, for $k = n$ and larger, the situation is less clear. The best known upper bound, which is $O(n)$ (and tight) for $k = n^{\alpha}$, $0 < \alpha < 1$ a constant, suddenly increases to $O(n \log n)$ for $k = n$, whereas the information-theoretic lower bound remains at $\Omega(n)$.

In this work, we prove that indeed there is a change of behavior around $k = n$. We show that for $k = \Theta(n)$, the random guessing strategy, and, in fact, any other nonadaptive strategy, cannot find the secret code with an expected number of less than $\Theta(n \log n)$ guesses. This can be proven via an entropy compression argument as used by Moser [2009] (see Theorem 4.1). For general k, our new lower bound for nonadaptive strategies is $\Omega(n \log(k)/\max\{\log(n/k), 1\})$. We also show that this lower bound is tight (up to constant factors). In fact, for $k \le n$, $O(n \log(k)/\max\{\log(n/k), 1\})$ random guesses suffice to determine the secret code. In other words, we extend Chvátal's result from $k \le n^{1-\varepsilon}$, $\varepsilon > 0$ a constant, to all $k \le n$.

The main contribution of our work is a (necessarily adaptive) strategy that for $k = n$ finds the secret code with only $O(n \log\log n)$ queries. This reduces the $\Theta(\log n)$ gap between the previous best upper and the lower bound to $\Theta(\log\log n)$. Like the previous strategies for $k \le n$, our new one does not use white answer pegs. Our strategy also improves the current best bounds for other values of k in the vicinity of n; see Theorem 2.1 for the precise result.

The central part of our guessing strategy is setting up suitable coin-weighing problems, solving them, and using the solution to rule out the possible occurrence of some colors at some positions. By a result of Grebinski and Kucherov [2000], these coin-weighing problems can be solved by relatively few independent random weighings.

Although our strategy thus is guided by probabilistic considerations, it can be derandomized to obtain a *deterministic* $O(n \log\log n)$ strategy for black-peg Mastermind with $k = n$ colors. Moreover, appealing to an algorithmic result of Bshouty [2009] instead of the result of Grebinski and Kucherov, we obtain a strategy that can be realized as a deterministic polynomial-time code-breaking algorithm.

We also improve the current best bounds for Mastermind with black and white answer pegs, which stand at $O(n \log n)$ for $n \le k \le n^2 \log\log n$. For these k, we prove that $O(n \log\log n)$ guesses suffice. We point out that this improvement is not an immediate consequence of our $O(n \log\log n)$ bound for $k = n$ black-peg Mastermind. Reducing the number of colors from k to n is a nontrivial subproblem as well. For example, when $k \ge n^{1+\varepsilon}$, Chvátal's strategy for the game with black and white answer pegs also uses $\Theta(n \log n)$ guesses to reduce the number of colors from k to n before employing a black-peg strategy to finally determine the secret code.

1.4. Related Work

Several results exist on the computational complexity of evaluating given guesses and answers. Stuckman and Zhang [2006] showed that it is NP-hard to decide whether there exists a secret code consistent with a given sequence of queries and black- and white-peg answers. This result was extended to black-peg Mastermind by Goodrich [2009b]. More recently, Viglietta [2012] showed that both hardness results also apply to the setting with only $k = 2$ colors. In addition, he proved that counting the number of consistent secret codes is #P-complete.

Another intensively studied question in the literature concerns the computation of (explicit) optimal winning strategies for small values of n and k. As described earlier, the foundation for these works was laid by Knuth's famous paper [Knuth 1977] for the case with $n = 4$ positions and $k = 6$ colors. His strategy is worst-case optimal. Koyama and Lai [1993] studied the average-case difficulty of Mastermind. They gave a strategy that solves Mastermind in an expected number of about 4.34 guesses if the secret string is sampled uniformly at random from all 6^4 possibilities, and they showed that this strategy is optimal. Today, a number of worst-case and average-case optimal winning strategies for different (small) values of n and k are known—both for the black- and white-peg version of the game [Goddard 2004; Jäger and Peczarski 2009] and for the black-peg version [Jäger and Peczarski 2011]. Nonadaptive strategies for specific values of n and k were studied in Goddard [2003].

In the field of computational intelligence, Mastermind is used as a benchmark problem. For several heuristics, among them genetic and evolutionary algorithms, it has been studied how well they play Mastermind [Kalisker and Camens 2003; Temporel and Kovacs 2003; Berghman et al. 2009; Guervós et al. 2011a, 2011b].

Trying to understand the intrinsic difficulty of a problem for such heuristics, Droste et al. [2006] suggested using a query complexity variant (called *black-box complexity*). For the so-called onemax test-function class, an easy benchmark problem in the field of evolutionary computation, the black-box complexity problem is just the Mastermind problem for two colors. This inspired, among others, the result [Doerr and Winzen 2014] showing that a memory-restricted version of Mastermind (using only two rows of the board) can still be solved in $O(n/\log n)$ guesses when the number of colors is constant.

Several privacy problems have been modeled via the Mastermind game. Goodrich [2009a] used black-peg Mastermind to study the extent of private genomic data leaked by comparing DNA sequences (even when using protocols only revealing the degree of similarity). Focardi and Luccio [2010] showed that certain API-level attacks on user PIN data can be seen as an extended Mastermind game.

1.5. Organization of This Article

We describe and analyze our $O(n \log\log n)$ strategy for $k = n$ colors in Section 2. In Section 3, we present a strong connection between the black pegs–only and the classic (black- and white-pegs) version of Mastermind. This yields, in particular, the claimed bound of $O(n \log\log n)$ for the classic version with $n \leq k \leq n^2 \log\log n$ colors. In Section 4, we analyze nonadaptive strategies. We prove a lower bound via entropy compression and show that it is tight for $k \leq n$ by extending Chavatal's analysis of random guessing to all $k \leq n$.

2. THE $O(n \log\log n)$ ADAPTIVE STRATEGY

In this section, we present the main contribution of this work, a black pegs–only strategy that solves Mastermind with $k = n$ colors in $O(n \log\log n)$ queries. We state our results for an arbitrary number $k = k(n)$ of colors; they improve on the previously known bounds for all $k = o(n \log n)$ with $k \geq n^{1-\varepsilon}$ for every fixed $\varepsilon > 0$.

Theorem 2.1. *For Mastermind with n positions and $k = k(n)$ colors, the following holds:*

—*If $k = \Omega(n)$, then there exists a randomized winning strategy that uses black pegs only and needs an expected number of $O(n \log\log n + k)$ guesses.*
—*If $k = o(n)$, then there exists a randomized winning strategy that uses black pegs only and needs an expected number of $O(n \log(\frac{\log n}{\log(n/k)}))$ guesses.*

The O-notation in Theorem 2.1 only hides absolute constants. Note that, setting $k =: n^{1-\delta}$, $\delta = \delta(n)$, the bound for $k = o(n)$ translates to $O(n \log(\delta^{-1}))$.

We describe our strategy and prove Theorem 2.1 in Sections 2.1 through 2.3. We discuss the derandomization of our strategy in Section 2.4.

2.1. Main Ideas

Our goal in this section is to give an informal sketch of our main ideas, and to outline how the $O(n \log\log n)$ bound for $k = n$ arises. For the sake of clarity, we nevertheless present our ideas in the general setting—it will be useful to distinguish between k and n notationally. As justified in Section 2.2.1, we assume that $k \leq n$ and that both k and n are powers of two.

A simple but crucial observation is that when we query a string $x \in [k]^n$ and the answer $\mathrm{eq}(z, x)$ is 0 (recall that z denotes Codemaker's secret color code), then we know that all queried colors are wrong for their respective positions (i.e., we have $z_i \neq x_i$ for all $i \in [n]$). To make use of this observation, we maintain, for each position i, a set $C_i \subseteq [k]$ of colors that we still consider possible at position i. Throughout our strategy, we reduce these sets successively, and once $|C_i| = 1$ for all $i \in [n]$, we have identified the secret code z. Variants of this idea have been used by several previous authors [Chvátal 1983; Goodrich 2009b].

Our strategy proceeds in phases. In each phase, we reduce the size of all sets C_i by a factor of two. Thus, before the jth phase, we will have $|C_i| \leq k/2^{j-1}$ for all $i \in [n]$. Consider now the beginning of the jth phase, and assume that all sets C_i have size exactly $k' := k/2^{j-1}$. Imagine that we query a random string r sampled uniformly from $C_1 \times \cdots \times C_n$. The expected value of $\mathrm{eq}(z, r)$ is n/k', and the probability that $\mathrm{eq}(z, r) = 0$ is $(1 - 1/k')^n \leq e^{-n/k'}$. If k' is significantly smaller than n, then this probability is very small, and we will not see enough 0-answers to exploit the simple observation that we made previously. However, if we group the n positions into $m := 4n/k'$ blocks of equal size $k'/4$, the expected contribution of each such block is 1/4, and the probability that a fixed such block contributes 0 to $\mathrm{eq}(z, r)$ is $(1 - 1/k')^{k'/4} \approx e^{-1/4}$ (i.e., constant). We will refer to blocks that contribute 0 to $\mathrm{eq}(z, r)$ as *0-blocks* in the following. For a random query, we expect a constant fraction of all m blocks to be 0-blocks. If we can identify which blocks these are, we can rule out a color at each position of each such block and make progress toward our goal.

As it turns out, the identification of the 0-blocks can be reduced to a *coin-weighing problem* that has been studied by several authors (see Grebinski and Kucherov [2000] and Bshouty [2000], and references therein). Specifically, we are given m coins of unknown integer weights and a spring scale. We can use the spring scale to determine the total weight of an arbitrary subset of coins in one weighing. Our goal is to identify the weight of every coin with as few weighings as possible.

In our setup, the "coins" are the blocks that we introduced earlier, and the "weight" of each block is its contribution to $\mathrm{eq}(z, r)$. To simulate weighings of subsets of coins by Mastermind queries, we use "dummy colors" for some positions (i.e., colors that we already know to be wrong at these positions). Using these, we can simulate the weighing of a subset of coins (=blocks) by copying the entries of the random query r in blocks that correspond to coins that we wish to include in our subset, and by using dummy colors for the entries of all other blocks.

Note that the *total weight* of our coins is $\mathrm{eq}(z, r)$. Typically, this value will be close to its expectation n/k' and therefore of the same order of magnitude as the number of blocks m. It follows from a coin-weighing result by Grebinski and Kucherov [2000] that $O(m/\log m)$ random queries (of the described block form, simulating the weighing of a random subset of coins) suffice to determine the contribution of each block to $\mathrm{eq}(z, r)$ with some positive probability. As observed previously, typically a constant fraction of all blocks contribute 0 to $\mathrm{eq}(z, r)$, and therefore we may exclude a color at a constant fraction of all n positions at this point.

Repeating this procedure of querying a random string r and using additional "random coin-weighing queries" to identify the 0-blocks eventually reduces the sizes of the sets C_i below $k'/2$, at which point the phase ends. In total, this requires $\Theta(k')$ rounds in which everything works out as sketched, corresponding to a total number of $\Theta(k' \cdot (m/\log m)) = \Theta(n/\log(4n/k'))$ queries for the entire phase.

Summing over all phases, this suggests that for $k = n$, a total number of

$$\sum_{j=1}^{\log k} O\left(\frac{n}{\log\left(4n/\left(\frac{k}{2^{j-1}}\right)\right)}\right) \stackrel{k=n}{=} O(n)\sum_{j=1}^{\log n}\frac{1}{j+1} = O(n\log\log n)$$

queries suffice to determine the secret code z, as claimed in Theorem 2.1 for $k = n$.

We remark that our precise strategy, Algorithm 1, slightly deviates from this description. This is due to a technical issue with our argument once the number k' of remaining colors drops below $C\log n$ for some $C > 0$. Specifically, beyond this point, the error bound that we derive for a fixed position is not strong enough to beat a union bound over all n positions. To avoid this issue, we stop our color reduction scheme before k' becomes that small (for simplicity, as soon as k' is less than $\sqrt{n}$) and solve the remaining Mastermind problem by asking random queries from the remaining set $C_1 \times \cdots \times C_n$, as originally proposed by Erdős and Rényi [1963] and Chvátal [1983].

2.2. Precise Description of Codebreaker's Strategy

2.2.1. Assumptions on n and k, Dummy Colors. Let us now give a precise description of our strategy. We begin by determining a dummy color for each position—that is, a color that we know to be wrong at that particular position. For this, we simply query the $n+1$ many strings $(1, 1, \ldots, 1), (2, 1, 1, \ldots, 1), \ldots, (2, 2, \ldots, 2) \in [k]^n$. Processing the answers to these queries in order, it is not hard to determine the location of all 1's and 2's in Codemaker's secret string z. In particular, this provides us with a dummy color for each position.

Next we argue that for the main part of our argument, we may assume that n and k are powers of two. To see this for n, note that we can simply extend Codemaker's secret string in an arbitrary way such that its length is the smallest power of two larger than n and pretend that we are trying to determine this extended string. To get the answers to our queries in this extended setting, we just need to add the contribution of the self-made extension part (which we determine ourselves) to the answers that Codemaker provides for the original string. As the extension changes n at most by a factor of two, our claimed asymptotic bounds are unaffected by this.

To argue that we may also assume k to be a power of two, we make use of the dummy colors that we already determined for the original value of k. Similar to the previous argument, we increase k to the next power of two and consider the game with this larger number of colors. To get the answers to our queries in this extended setting from Codemaker (who still is in the original setting), it suffices to replace every occurrence of a color that is not in the original color set with the dummy color at the respective position.

We may and will also assume that $k \le n$. If $k > n$, we can trivially reduce the number of colors to n by making k monochromatic queries. With this observation, the first part of Theorem 2.1 follows immediately from the $O(n\log\log n)$ bound that we prove for the case $k = n$.

2.2.2. Eliminating Colors With Coin-Weighing Queries. With these technicalities out of the way, we can focus on the main part of our strategy. As sketched earlier, our strategy operates in *phases*, where in the jth phase we reduce the sizes of the sets C_i from $k/2^{j-1}$ to $k/2^j$. For technical reasons, we do not allow the sizes of C_i to drop below $k/2^j$ during phase j—for instance, once we have $|C_i| = k/2^j$ for some position $i \in [n]$, we no longer remove colors from C_i at that position and ignore any information that would allow us to do so.

Each phase is divided into a large number of *rounds*, where a round consists of querying a random string r and subsequently identifying the 0-blocks (blocks that contribute 0 to $\text{eq}(z,r)$) by the coin-weighing argument outlined earlier.

To simplify the analysis, the random string r is sampled from the same distribution throughout the entire phase. Specifically, at the beginning of phase j, we define the set $\mathcal{R}_j := C_1 \times \cdots \times C_n$ and sample the random string r uniformly at random from $\mathcal{R}_j$ in each round of phase j. Note that we do not adjust $\mathcal{R}_j$ during phase j; information about excluded colors that we gain during phase j will only be used in the definition of the set $\mathcal{R}_{j+1}$ in phase $j+1$.

We now introduce the formal setup for the coin-weighing argument. As before, we let $k' = k/2^{j-1}$ and partition the n positions into $m := 4n/k'$ blocks of size $k'/4$. More formally, for every $s \in [m]$ we let $B_s := \{(s-1)k'/4+1, \ldots, sk'/4\}$ denote the indices of block s, and we denote by $v_s := |\{i \in B_s : z_i = r_i\}|$ the contribution of block B_s to $\text{eq}(z,r)$. (Note that $\sum_{s\in[m]} v_s = \text{eq}(z,r)$.) As indicated previously, we wish to identify the 0-blocks—that is, the indices $s \in [m]$ for which $v_s = 0$.

For $y \in \{0,1\}^m$, define r_y as the query that is identical to r on the blocks B_s for which $y_s = 1$, and identical to the string of dummy colors on all other blocks. Thus, $\text{eq}(z, r_y) = \sum_{s\in[m], y_s=1} v_s$. With this observation, identifying the values v_s from a set of queries of form r_y is equivalent to a coin-weighing problem in which we have m coins with positive integer weights that sum up to $\text{eq}(z,r)$: Querying r_y in the Mastermind game provides exactly the information that we obtain from weighing the set of coins indicated by y.

We will only bother with the coin weighing if the initial random query of the round satisfies $\text{eq}(z,r) \le m/2$. (Recall that the expected value of $\text{eq}(z,r)$ is $m/4$.) If this is the case, we query an appropriate number $f(m)$ of strings of form r_y, with $y \in \{0,1\}^m$ sampled uniformly at random and independently. The function $f(m)$ is implicit in the proof of the coin-weighing result of Grebinski and Kucherov [2000]; it is in $\Theta(m/\log m)$ and guarantees that the coin weighing succeeds with probability at least 1/2. Thus, with probability at least 1/2, these queries determine all values v_s and, in particular, identify all 0-blocks. Note that the inequality $\text{eq}(z,r) \le m/2$ also guarantees that at least half of the m blocks are 0-blocks.

We say that a round is *successful* if $\text{eq}(z,r) \le m/2$ and if the coin weighing successfully identifies all 0-blocks. In each successful round, we update the sets C_i as outlined previously—that is, for each position i that is in a 0-block and for which $|C_i| > k'/2$, we set $C_i := C_i \setminus \{r_i\}$. Note that it might happen that r_i is a color that was already removed from C_i in an earlier round of the current phase, in which case C_i remains unchanged. If a round is unsuccessful, we do nothing and continue with the next round.

This completes the description of our strategy for a given phase. We abandon this color reduction scheme once k' is less than $\sqrt{n}$. At this point, we simply ask queries sampled uniformly and independently at random from the current set $\mathcal{R} = C_1 \times \cdots \times C_n$. We do so until the answers uniquely determine the secret code z. It follows from the result of Chvátal [1983] that the expected number of queries needed for this is $O(n \log k' / \log(n/k')) = O(n)$.

This concludes the description of our strategy. It is summarized in Algorithm 1. Correctness is immediate from our discussion, and it remains to bound the expected number of queries that the strategy makes.

2.3. Proof of Theorem 2.1

We begin by bounding the expected number of rounds in the jth phase.

CLAIM 2.2. *The expected number of rounds required to complete phase j is $O(k') = O(k/2^j)$.*

ALGORITHM 1: Playing Mastermind With Many Colors

```
Determine a dummy color for each position;
foreach $i \in [n]$ do $C_i \leftarrow [k]$;
;
$j \leftarrow 0$ and $k' \leftarrow k$;
while $k' > \sqrt{n}$ do
    $j \leftarrow j+1$, $k' \leftarrow k/2^{j-1}$, $\mathcal{R}_j \leftarrow C_1 \times \cdots \times C_n$, and $m \leftarrow 4n/k'$;
    repeat
        Select a string $r$ uniformly at random from $\mathcal{R}_j$ and query eq$(z, r)$;
        if eq$(z, r) \le m/2$ then
            for $i = 1, \ldots, f(m)$   /* $f(m) = \Theta(m/\log m)$ /* do
                Sample $y$ uniformly at random from $\{0, 1\}^m$ and query eq$(z, r_y)$;
            if these $f(m)$ queries determine the 0-blocks of $r$ then
                foreach $i \in [n]$ do
                    if $i$ is in a 0-block and $|C_i| > k'/2$ then $C_i \leftarrow C_i \setminus \{r_i\}$;
                    ;
    until $\forall i \in [n] : |C_i| = k'/2$;
$\mathcal{R} \leftarrow C_1 \times \cdots \times C_n$;
Select strings $r$ independently and uniformly at random from $\mathcal{R}$, and query eq$(z, r)$ until $z$
is determined;
```

Proof. We first show that a round is successful with probability at least 1/4. Recall that eq(z, r) has an expected value of $n/k' = m/4$. Thus, by Markov's inequality, we have eq$(z, r) \le m/2$ with probability at least 1/2. Moreover, as already mentioned, the proof of the coin-weighing result by Grebinski and Kucherov [2000] implies that our $f(m) = \Theta(m/\log m)$ random coin-weighing queries identify all 0-blocks with probability at least 1/2. Thus, in total, the probability for a successful round is at least $1/2 \cdot 1/2 = 1/4$.

We continue by showing that the probability that a successful round decreases the number of available colors for a fixed position, say position 1, is at least 1/4. Note that this happens if $r \in \mathcal{R}_j$ satisfies the following two conditions:

(i) $v_1 = 0$ (i.e., block B_1 is a 0-block with respect to r), and
(ii) $r_1 \in C_1$ (i.e., the color r_1 has not been excluded from C_1 in a previous round of phase j).

For (i), recall that in a successful round, at least $m/2$ of the m blocks are 0-blocks. It follows by symmetry that B_1 is a 0-block with probability at least 1/2. Moreover, conditional on (i), r_1 is sampled uniformly at random from the $k' - 1$ colors that are different from z_1 and were in C_1 at the beginning of the round. Thus, the probability that r_1 is in the current set C_1 is $|C_1|/(k' - 1)$, which is at least 1/2 because we do not allow $|C_1|$ to drop below $k'/2$. We conclude that, conditional on a successful round, the random query r decreases $|C_1|$ by one with probability at least $1/2 \cdot 1/2 = 1/4$.

Thus, in total, the probability that a round decreases $|C_1|$ by one is at least $1/4 \cdot 1/4 = 1/16$ throughout our strategy. It follows that the probability that after t successful rounds in phase j we still have $|C_1| > k'/2$ is bounded by the probability that in t independent Bernoulli trials with success probability 1/16, we observe fewer than $k'/2$ successes. If $t/16 \ge k'$, by Chernoff bounds, this probability is bounded by e^{-ct} for some absolute constant $c > 0$.

Let us now denote the number of rounds that phase j takes by the random variable T. By a union bound, the probability that $T \ge t$ (i.e., that after t steps at one of the positions $i \in [n]$ we still have $|C_i| > k'/2$) is bounded by ne^{-ct} for $t \ge 16k'$. It follows that

$$\mathbb{E}[T] = \sum_{t \geq 1} \Pr[T \geq t] \leq 16k' + n \sum_{t > 16k'} e^{-ct} = 16k' + ne^{-\Omega(k')} = O(k'), \tag{1}$$

where the last step is due to $k' \geq \sqrt{n} = \omega(\log n)$. □

With Claim 2.2 in hand, we can bound the total number of queries required throughout our strategy by a straightforward calculation.

Proof of Theorem 2.1. Recall that for each phase j, we have $m = \Theta(n/k') = \Theta(n/(k/2^{j-1}))$ and that $f(m) = \Theta(m/\log m)$. Thus, by Claim 2.2, the expected number of queries that our strategy asks in phase j is bounded by

$$O(k') \cdot (1 + f(m)) = O\left(\frac{n}{\log\left(\frac{n}{k/2^j}\right)}\right) = O\left(\frac{n}{\log(n/k) + j}\right).$$

It follows that throughout the main part of our strategy, we ask an expected number of queries of at most

$$O(n) \sum_{j=1}^{\log k} \frac{1}{\log(n/k) + j} = O(n(\log\log n - \log\log(n/k))) = O\left(n \log\left(\frac{\log n}{\log(n/k)}\right)\right).$$

(This calculation is for $k < n$; as observed before, for $k = n$, a very similar calculation yields a bound of $O(n \log\log n)$.) As the number of queries for determining the dummy colors and for wrapping up at the end is only $O(n)$, Theorem 2.1 follows. □

2.4. Derandomization

The strategy that we presented in the previous section can be derandomized and implemented as a polynomial-time algorithm.

Theorem 2.3. *The bounds stated in Theorem 2.1 can be achieved by a deterministic winning strategy. Furthermore, this winning strategy can be realized in polynomial time.*

Proof. The main loop of the algorithm described earlier uses randomization in two places: for generating the random string r of each round (line 8 in Algorithm 1), and for generating the $f(m)$ many random coin-weighing queries r_y used to identify the 0-blocks of r if $\mathrm{eq}(z, r) \leq m/2$ (line 11).

The derandomization of the coin-weighing algorithm is already given in the work of Grebinski and Kucherov [2000]. They showed that a set of $f'(m) = \Theta(m/\log m)$ random coin-weighing queries $y^1, \dots, y^{f'(m)}$, sampled from $\{0, 1\}^m$ independently and uniformly at random, has, with some positive probability, the property that it distinguishes any two distinct coin-weighing instances in the following sense: for any two distinct vectors v, w with nonnegative integer entries such that $\sum_{s\in[m]} v_s \leq m/2$ and $\sum_{s\in[m]} w_s \leq m/2$, there exists an index $j \in [f'(m)]$ for which $\sum_{s\in[m], y^j_s=1} v_s \neq \sum_{s\in[m], y^j_s=1} w_s$. It follows by the probabilistic method that, deterministically, there is a set $D \subseteq \{0, 1\}^m$ of size at most $f'(m)$ such that the answers to the corresponding coin-weighing queries identify every possible coin-weighing instance. Hence, we can replace the $f(m)$ random coin-weighing queries of each round by the $f'(m)$ coin-weighing queries corresponding to the fixed set D.

It remains to derandomize the choice of r in each round. As before, we consider $m := 4n/k'$ blocks of size $k'/4$, where k' is the size of the sets C_i at the beginning of a phase. To make sure that a constant fraction of all queries in a phase satisfy $\mathrm{eq}(z, r) \leq m/2$ (compare line 9 of Algorithm 1), we ask a set of k' queries such that for each position $i \in [n]$, every color in C_i is used at position i in exactly one of these queries. (If all sets C_i are equal, this can be achieved by simply asking k' monochromatic queries.) The sum of all returned scores must be exactly n, and therefore we cannot get a score of more than $m/2 = 2n/k'$ for more than $k'/2$ queries. In this way, we ensure that for at least $k - k'/2 = k'/2$ queries, we get a score of at most $m/2$.

As in the randomized version of our strategy, in each of these $k'/2$ queries at least half of the blocks must be 0-blocks. We can identify those by the derandomized coin-weighing discussed previously. Now consider a fixed block. As it has size $k'/4$, it can be a *non*-0-block in at most $k'/4$ queries. Thus, it is a 0-block in at least $k'/2 - k'/4 = k'/4$ of the queries.

To summarize, we have shown that by asking k' queries of the preceding form, we get at least $k'/2$ queries of score at most $m/2$. For each of them, we identify the 0-blocks by coin-weighing queries. This allows us to exclude at least $k'/4$ colors at each position. For instance, as in the randomized version of our strategy, we can reduce the number of colors by a constant factor using only $O(k' \cdot m/\log m) = O(n/\log(4n/k))$ queries. By similar calculations as before, the same asymptotic bounds follow.

We abandon the color reduction scheme when k' is a constant. At this point, we can solve the remaining problem in time $O(n)$ by repeatedly using the argument that we used to determine the dummy colors in Section 2.2.1.

Note that all that we have described previously can easily be implemented in polynomial time if we can solve the coin-weighing subproblems in polynomial time. An algorithm for doing the latter is given in the work of Bshouty [2009]. Using this algorithm as a building block, we obtain a deterministic polynomial-time strategy for Codebreaker that achieves the bounds stated in Theorem 2.1. □

3. MASTERMIND WITH BLACK AND WHITE ANSWER PEGS

In this section, we analyze the Mastermind game in the classic version with both black and white answer pegs. Interestingly, there is a strong general connection between the two versions. Roughly speaking, we can use a strategy for the $k = n$ black-peg game to learn which colors actually occur in the secret code of a black/white-peg game with n positions and n^2 colors. Having thus reduced the number of relevant colors to at most n, Codebreaker can again use a $k = n$ black-peg strategy (ignoring the white answer pegs) to finally determine the secret code.

More precisely, for all $k, n \in \mathbb{N}$, let us denote by $b(n, k)$ the minimum (taken over all strategies) maximum (taken over all secret codes) expected number of queries needed to find the secret code in a black-peg Mastermind game with k colors and n positions. Similarly, denote by $bw(n, k)$ the corresponding number for the game with black and white answer pegs. Then we show the following.

THEOREM 3.1. *For all $k, n \in \mathbb{N}$ with $k \geq n$,*

$$bw(n, k) = \Theta(k/n + b(n, n)).$$

Combining this with Theorem 2.1, we obtain a bound of $O(n \log\log n)$ for black/white Mastermind with $n \leq k \leq n^2 \log\log n$ colors, improving all previous bounds in that range.

For the case $k \leq n$, it is not hard to see that $bw(n, k) = \Theta(b(n, k))$ (see Corollary 3.3). Together with Theorem 3.1, this shows that to understand black/white-peg Mastermind for all n and k, it suffices to understand black-peg Mastermind for all n and k.

Before proving Theorem 3.1, let us derive a few simple preliminary results on the relation of the two versions of the game.

LEMMA 3.2. *For all n, k,*

$$bw(n,k) \geq b(n,k) - k + 1.$$

PROOF. We show that we can simulate a strategy in the black/white Mastermind game by one receiving only black-pegs answers and using $k-1$ more guesses. Fix a strategy for black/white Mastermind. Our black-peg strategy first asks $k-1$ monochromatic queries. This tells us how often each of the k color arises in the secret code. From now on, we can play the strategy for the black/white game. Although we only receive black answer pegs, we can compute the number of white pegs that we would have gotten in the black/white game from the just obtained information on how often each color occurs in the code. With this information available, we can indeed play as in the given strategy for black/white Mastermind. □

Lemma 3.2 will be used to prove that the $b(n, n)$ term in the statement of Theorem 3.1 cannot be avoided. As a corollary, it yields that white answer pegs are not extremely helpful when $k = O(n)$.

COROLLARY 3.3. *For all $k \leq n$,*

$$bw(n,k) = \Theta(b(n,k)).$$

PROOF. Obviously, $bw(n,k) \leq b(n,k)$ for all n, k. If $k = o(n)$, then the information-theoretic lower bound $b(n,k) = \Omega(n \log k / \log n)$ is of larger order than k, and hence the preceding lemma shows the claim. For $k = \Theta(n)$, note first that both $b(n,k)$ and $bw(n,k)$ are in $\Omega(n)$ due to the information-theoretic argument. If $b(n,k) = O(n)$, there is nothing to show. If $b(n,k) = \omega(n)$, we again invoke Lemma 3.2. □

In the remainder of this section, we prove Theorem 3.1. To describe the upper bound, let us fix the following notation. Let C be the set of all available colors and $k = |C|$. Denote by $z \in C^n$ the secret code chosen by Codemaker. Denote by $C^* := \{z_i \mid i \in [n]\}$ the (unknown) set of colors in z.

Codebreaker's strategy leading to the bound of Theorem 3.1 consists of roughly these three steps:

(1) Codebreaker first asks roughly k/n guesses containing all colors. Only colors in a guess receiving a positive answer can be part of the secret code, so this reduces the number of colors to be regarded to at most n^2. In addition, Codebreaker can learn from the answers the cardinality n' of C^*—that is, the number of distinct colors in the secret code.
(2) By asking an expected number of $\Theta(n')$ (dependent) random queries, Codebreaker learns n' disjoint sets of colors of size at most n such that each color of C^* is contained in exactly one of these sets. Denote by k' the cardinality of the largest of these sets.
(3) Given such a family of sets, Codebreaker can learn C^* with an expected number of $b(n', k')$ queries by simulating an optimal black-peg Mastermind strategy. Once C^* is known, an expected number of $b(n, n')$ queries determine the secret code, using an optimal black-peg strategy for n' colors.

Each of these steps is made precise in the following. Before doing so, we remark that after a single query, Codebreaker may detect $|C^* \cap X|$ for any set X of at most n colors via a single Mastermind query to be answered by black and white answer pegs.

LEMMA 3.4. *For an arbitrary set X of at most n colors, let* $\mathrm{col}(X) := |C^* \cap X|$*, the number of colors of X occurring in the secret code. After a single initial query, Codebreaker can learn* $\mathrm{col}(X)$ *for any X via a single Mastermind query to be answered by black and white pegs.*

Proof. As the single initial query, Codebreaker may ask $(1,\dots,1)$, the code consisting of color 1 only. Denote by b the number of black pegs received (there cannot be a white answer peg). This is the number of occurrences of color 1 in the secret code.

Let $X \subseteq C$, $\nu := |X| \leq n$. To learn $\mathrm{col}(X)$, Codebreaker extends X to a multiset of n colors by adding the color 1 exactly $n-\nu$ times and guesses a code arbitrarily composed of this multiset of colors. Let y be the total number of (black and white) answer pegs received. Then $\mathrm{col}(X) = y - \min\{n-\nu, b\}$, if $1 \notin X$ or $b = 0$, and $\mathrm{col}(X) = y - \min\{n-\nu, b-1\}$ otherwise. $\square$

To ease the language, we shall call a query determining $\mathrm{col}(X)$ a *color query*. We now show that using roughly k/n color queries, Codebreaker can learn the number $|C^*|$ of different colors occurring in the secret code and exclude all but $n|C^*|$ colors.

Lemma 3.5. *With $\lceil k/n \rceil$ color queries, Codebreaker can learn both $|C^*|$ and a superset C_0 of C^* consisting of at most $n|C^*|$ colors.*

Proof. Let $X_1,\dots,X_{\lceil k/n \rceil}$ be a partition of C into sets of cardinality at most n. By asking the corresponding $\lceil k/n \rceil$ color queries, Codebreaker immediately learns $|C^*| := \sum_{i=1}^{\lceil k/n \rceil} \mathrm{col}(X_i)$. Additionally, $C_0 := \bigcup\{X_i \mid \mathrm{col}(X_i) > 0\}$ is the desired superset. $\square$

Lemma 3.6. *Assume that Codebreaker knows the number $n' = |C^*|$ of different colors in z as well as a set $C_0 \supseteq C^*$ of colors such that $|C_0| \leq n|C^*|$.*

Then with an expected number of $\Theta(n')$ color queries, Codebreaker can find a family $C_1,\dots,C_{n'}$ of disjoint subsets of C_0, each of size at most $\lceil |C_0|/n' \rceil \leq n$, such that $C^* \subseteq C_1 \cup \dots \cup C_{n'}$ and $|C^* \cap C_i| = 1$ for all $i \in [n']$.

Proof. Roughly speaking, Codebreaker's strategy is to ask color queries having an expected answer of one. With constant probability, such a query contains exactly one color from C^*. This strategy is formalized by Algorithm 2.

ALGORITHM 2: Codebreaker's Strategy

while $n' > 0$ **do**
  $k' \leftarrow \lceil |C_0|/n' \rceil$;
  Let $C_{n'}$ be a random subset of C with $|C_{n'}| = k'$;
  Ask the color query $C_{n'}$;
  if $\mathrm{col}(C_{n'}) = 1$ **then**
    $C_0 \leftarrow C_0 \setminus C_{n'}$;
    $n' \leftarrow n' - 1$;

For the analysis, note first that the value of k' during the application of the preceding strategy does not increase. In particular, all sets C_i defined and queried have cardinality at most $\lceil |C_0|/n' \rceil \leq n$. It is also clear that the preceding strategy constructs a sequence of disjoint C_i and that for each color occurring in z there is exactly one C_i containing this color.

It remains to prove the estimate on the expected number of queries. To this aim, we first note that throughout a run of this strategy, n' is the number of colors of C^* left in C_0. Hence, the event "$\mathrm{col}(C_{n'}) = 1$" occurs with probability

$$\frac{n'k'(|C_0|-n')\dots(|C_0|-n'-k'+2)}{|C_0|\dots(|C_0|-k'+1)} \geq \frac{(|C_0|-n')\dots(|C_0|-n'-k'+2)}{(|C_0|-1)\dots(|C_0|-k'+1)}$$
$$\geq \left(\frac{|C_0|-n'-k'+2}{|C_0|-k'+1}\right)^{k'-1} = \left(1-\frac{n'-1}{|C_0|-k'+1}\right)^{k'-1}$$

$$\geq \left(1 - \frac{n' - 1}{|C_0| - (|C_0|/n')}\right)^{k'-1}$$
$$\geq \left(1 - \frac{|C_0|/(k' - 1)}{|C_0| - (|C_0|/n')}\right)^{k'-1}$$
$$\geq \left(1 - \frac{1}{(k' - 1)(1 - 1/n')}\right)^{k'-1},$$

which is bounded from below by a constant (the later estimates assume $n' \geq 2$; for $n' = 1$, the second term of the sequence of inequalities already is one).

Consequently, with constant probability, the randomly chosen $C_{n'}$ satisfies "$\mathrm{col}(C_{n'}) = 1$". Hence, after an expected constant number of iterations of the while-loop, such a $C_{n'}$ will be found. Since each such success reduces the value of n' by one, a total expected number of $\Theta(|C^*|)$ iterations suffices to find the desired family of sets $(C_i)_{i\in[n']}$. □

Given a family of sets as just constructed, Codebreaker can simulate a black-peg strategy to determine C^*.

LEMMA 3.7. *Let $C_1, \ldots, C_{n'}$ be a family of disjoint subsets of C such that $C^* \subseteq C_1 \cup \ldots \cup C_{n'}$ and $|C^* \cap C_i| = 1$ for all $i \in [n']$. Assume that $k' := \max\{|C_i| \mid i \in [n']\} \leq n$. Then Codebreaker can detect C^* using an expected number of $b(n', k')$ color queries.*

PROOF. Let $z' \in C_1 \times \cdots \times C_{n'}$ be the unique such string consisting of colors in C^* only. Note that in black-peg Mastermind, the particular sets of colors used at each position are irrelevant. Hence, there is a strategy for Codebreaker to detect z' using an expected number of $b(n', k')$ guesses from $C_1 \times \cdots \times C_{n'}$ and receiving black-peg answers only.

We now show that for each such query, there is a corresponding color query in the (n, k) black/white Mastermind game giving the same answer. Hence, we may simulate the black-peg game searching for z' by such color queries. Since z' contains all colors of C^* and no other colors, once found, it reveals the set of colors occurring in the original secret code z.

Let $y' \in C_1 \times \cdots \times C_{n'}$ be a query in the black-peg Mastermind game searching for z'. For each position $i \in [n']$, we have $z'_i = y'_i$ if and only if $y'_i \in C_i$ is the unique color from C_i that is in C^*. As moreover the sets $(C_i)_{i\in[n']}$ are disjoint, we have $\mathrm{eq}(z', y') = \mathrm{col}(\{y'_1, \ldots, y'_{n'}\})$, and we can obtain this value (i.e., the black-peg answer for the guess y' relative to z') by a color query relative to z. □

Note that if our only goal is to find out C^*, then for $k \ll n^2$ we can be more efficient by asking more color queries in Lemma 3.5, leading to a smaller set C_0, to smaller sets C_i in Lemma 3.6, and thus to a smaller k' value in Lemma 3.7. Since this will not affect the asymptotic bound for the total numbers of queries used in the black/white-peg game, we omit the details.

PROOF OF THEOREM 3.1. The upper bound follows easily from applying Lemmas 3.4 through 3.7, which show that Codebreaker can detect the set C^* of colors arising in the secret code z with an expected number of $1 + \lceil k/n \rceil + O(n) + b(n, n)$ guesses. Since $|C^*| \leq n$, he can now use a strategy for black-peg Mastermind and determine z with another expected number of $b(n, n)$ guesses. Note that $b(n, n) = \Omega(n)$, so this proves the upper bound.

We argue that this upper bound is optimal apart from constant factors. Assume first that the secret code is a random monochromatic string (Codemaker may even announce this). Fix a (possibly randomized) strategy for Codebreaker. With probability at least

1/2, this strategy does not use the particular color in any of the first $k/(2n)$ guesses. It then also did not guess the correct code. Hence, the expected number of queries necessary to find the code is at least $k/(4n)$.

We finally show that for $k \geq n$, also the $b(n, n)$ term cannot be avoided. By the information-theoretic argument, there is nothing to show if $b(n, n) = \Theta(n)$. Hence, assume that $b(n, n) = \omega(n)$. We will show that $bw(n, k) + n + 1 \geq bw(n, n)$. The claim then follows from $bw(n, n) = \Theta(b(n, n))$ (Corollary 3.3).

We show that we can solve the $k = n$ color black/white Mastermind game by asking $n + 1$ preliminary queries and then simulating a strategy for black/white Mastermind with n positions and $k > n$. As in Section 2.2.1, we use $n + 1$ queries to learn for each position whether it has color 1 or not. We then simulate a given strategy for $k > n$ colors as follows. In a k-color query, replace all colors greater than n by color 1. Since we know the positions of the pegs in color 1, we can reduce the answers by the contribution of these additional 1-pegs in the query. This gives the answer that we would have gotten in reply to the original query (since the secret code does not contain colors larger than n). Consequently, we can now simulate the k-color black/white Mastermind strategy in an n-color black/white Mastermind game. □

4. NONADAPTIVE STRATEGIES

When analyzing the performance of nonadaptive strategies, it is not very meaningful to ask for the number of queries needed until the secret code is queried for the first time. Instead, we ask for the number of queries needed to identify it.

In their work on the 2-color black-peg version of Mastermind, Erdős and Rényi [1963] showed that random guessing needs, with high probability, $(2 + o(1))n/\log n$ queries to identify the secret code, and that this is in fact best possible among nonadaptive winning strategies. The upper bound was derandomized by Lindström [1964, 1965] and independently by Cantor and Mills [1966]. In other words, for 2-color black-pegs Mastermind, a deterministic nonadaptive winning strategy using $(2 + o(1))n/\log n$ guesses exists, and no nonadaptive strategy can do better.

For adaptive strategies, only a weaker lower bound of $(1+o(1))n/\log n$ is known. This bound results from the information-theoretic argument mentioned in Section 1.2. It remains a major open problem whether there exists an adaptive strategy that achieves this bound. In fact, it is not even known whether adaptive strategies can outperform the random guessing strategy by any constant factor.

In this section, we prove that for Mastermind with $k = \Theta(n)$ colors, adaptive strategies are indeed more powerful than nonadaptive ones, and they outperform them even in order of magnitude. More precisely, we show that any nonadaptive strategy needs $\Omega(n \log n)$ guesses. Since we know from Section 2 that adaptively we can achieve a bound of $O(n \log \log n)$, this separates the performance of nonadaptive strategies from that of adaptive ones. Our result answers a question left open in Goddard [2003].

The $\Omega(n \log n)$ bound for nonadaptive strategies is tight. As we will show in Theorem 4.3, there exists a deterministic nonadaptive strategy that achieves the bound up to constant factors.

4.1. Lower Bound for Nonadaptive Strategies

For the formal statement of the bound, we use the following notation. A deterministic nonadaptive strategy is a fixed ordering $x^1, x^2, \ldots, x^{k^n}$ of all possible guesses (i.e., the elements of $[k]^n$). A randomized nonadaptive strategy is a probability distribution over such orderings. For a given secret code $z \in [k]^n$, we ask for the smallest index j such that the queries $x^1, \ldots, x^j$ together with their answers $\mathrm{eq}(z, x^1), \ldots, \mathrm{eq}(z, x^j)$ uniquely determine z.

Mastermind with nonadaptive strategies is also referred to as static Mastermind [Goddard 2003].

THEOREM 4.1. *For any (randomized or deterministic) nonadaptive strategy for black-peg Mastermind with n positions and k colors, the expected number of queries needed to determine a secret code z sampled uniformly at random from $[k]^n$ is $\Omega(\frac{n \log k}{\max\{\log(n/k), 1\}})$.*

Theorem 4.1 shows, in particular, that for any nonadaptive strategy, there exists a secret code $z \in [k]^n$ that can only be identified after $\Omega(n \log k / \max\{\log(n/k), 1\})$ queries. For $k \geq n$, this is an improvement of $\Theta(\log n)$ over the information-theoretic lower bound mentioned in Section 1. For the case $k = \Theta(n)$, Theorem 4.1 gives a lower bound of $\Omega(n \log n)$ guesses for every nonadaptive strategy, showing that adaptive strategies are indeed more powerful than nonadaptive ones in this regime (recall Theorem 2.1).

To give an intuition for the correctness of Theorem 4.1, note that for a uniformly chosen secret code $z \in [k]^n$, for any single fixed guess x of a nonadaptive strategy the answer $\mathrm{eq}(z, x)$ is binomially distributed with parameters n and $1/k$. In other words, $\mathrm{eq}(z, x)$ will typically be within the interval $n/k \pm O(\sqrt{n/k})$. Hence, we can typically encode the answer using $\log(O(\sqrt{n/k})) = O(\log(n/k))$ bits. Stated differently, our "information gain" is usually $O(\log(n/k))$ bits. Since the secret code "holds $n \log k$ bits of information," we would expect that we have to make $\Omega(n \log k / \log(n/k))$ guesses.

To turn this intuition into a formal proof, we recall the notion of entropy: for a discrete random variable Z over a domain D, the *entropy* of Z is defined by $H(Z) := -\sum_{z \in D} \Pr[Z = z] \log(\Pr[Z = z])$. Intuitively speaking, the entropy measures the amount of information that the random variable Z carries. For example, if Z corresponds to a random coin toss with Pr['heads'] = Pr['tails'] = 1/2, then Z carries 1 bit of information. However, a biased coin toss with Pr['heads'] = 2/3 carries less (roughly 0.918 bits) information, as we know that the outcome of heads is more likely. In our proof, we use the following properties of the entropy, which can easily be seen to hold for any two random variables Z, Y over domains D_Z, D_Y:

(E1) If Z is determined by the outcome of Y (i.e., $Z = f(Y)$ for a deterministic function f), then we have $H(Z) \leq H(Y)$.
(E2) We have $H((Z, Y)) \leq H(Z) + H(Y)$.

The inequality in (E2) holds with equality if and only if the two variables Z and Y are independent.

PROOF OF THEOREM 4.1. Next we show that there is a time $s = \Omega(\frac{n \log k}{\max\{\log(n/k), 1\}})$ such that any deterministic strategy at any time earlier than s determines less than half of the secret codes. Consequently, any deterministic strategy needs an expected time of at least $s/2$ to determine a secret chosen uniformly at random. Since any randomized strategy is a convex combination of deterministic ones, this latter statement also holds for randomized strategies.

Let $S = (x^1, x^2, \dots)$ denote a deterministic strategy of Codebreaker. We first show a lower bound on the number of guesses that are needed to identify at least half of all possible secret codes. For $j = 1, \dots, k^n$, let $A_j = A_j(S) \subseteq [k]^n$ denote the set of codes that can be uniquely determined from the answers to the queries $x^1, \dots, x^j$. Let s be the smallest index for which $|A_s| \geq k^n/2$.

Consider a code $Z \in [k]^n$ sampled uniformly at random, and set $Y_i := \mathrm{eq}(Z, x^i)$, $1 \leq i \leq s$. Moreover, let

$$\tilde{Z} = \begin{cases} Z & \text{if } Z \in A_s, \\ \text{'fail'} & \text{if } Z \notin A_s. \end{cases}$$

By our definitions, the sequence $Y := (Y_1, Y_2, \ldots, Y_s)$ determines $\tilde{Z}$, and hence by (E1) we have

$$H(\tilde{Z}) \leq H(Y). \tag{2}$$

Moreover, we have

$$\begin{aligned} H(\tilde{Z}) &= -\sum_{z \in A_s} \Pr[\tilde{Z} = z] \log(\Pr[\tilde{Z} = z]) - \Pr[\tilde{Z} = \text{`fail'}] \log(\Pr[\tilde{Z} = \text{`fail'}]) \\ &\geq -\sum_{z \in A_s} \Pr[Z = z] \log(\Pr[Z = z]) \\ &= \frac{|A_s|}{k^n} \log(k^n) \\ &\geq \frac{1}{2} n \log k. \end{aligned} \tag{3}$$

We now derive an upper bound on $H(Y)$. For every i, Y_i is binomially distributed with parameters n and $1/k$. Therefore, its entropy is (e.g., see Jacquet and Szpankowski [1999])

$$H(Y_i) = \frac{1}{2} \log \left(2\pi e \frac{n}{k} \left(1 - \frac{1}{k} \right) \right) + \frac{1}{2} + O\left(\frac{1}{n} \right) = O(\max\{\log(n/k), 1\}).$$

We thus obtain

$$H(Y) \overset{\text{(E2)}}{\leq} \sum_{i=1}^{s} H(Y_i) = sH(Y_1) = sO(\max\{\log(n/k), 1\}). \tag{4}$$

Combining (2), (3), and (4), we obtain

$$s = \Omega\left(\frac{n \log k}{\max\{\log(n/k), 1\}} \right).$$

Since by definition of s at least half of all secret codes in $[k]^n$ can only be identified by the strategy S after at least s guesses, it follows that the expected number of queries needed to identify a uniformly chosen secret code is at least $s/2$. □

4.2. Upper Bound for Nonadaptive Strategies

We first show that for $k = \Theta(n)$, a random guessing strategy asymptotically achieves the lower bound from Theorem 4.1. Afterward, we show that one can also derandomize this.

LEMMA 4.2. *For black-peg Mastermind with n positions and $k = \Theta(n)$ colors, the random guessing strategy needs an expected number of $O(n \log n)$ queries to determine an arbitrary fixed code $z \in [k]^n$. Furthermore, for a large enough constant C, $Cn \log n$ queries suffice with probability $1 - o(1)$.*

PROOF. We can easily eliminate colors whenever we receive a 0-answer. For every position $i \in [n]$, we need to eliminate $k - 1$ potential colors. This can be seen as having n parallel coupon collectors, each of which needs to collect $k - 1$ coupons.

The probability that for a random guess we get an answer of 0 is $(1 - 1/k)^n$ (i.e., constant). Conditional on a 0-answer, the color excluded at each position is sampled uniformly from all $k - 1$ colors that are wrong at that particular position. Thus, the probability that at least one of the $k - 1$ wrong colors at one fixed position is not eliminated by the first t 0-answers is bounded by $(k-1)(1 - \frac{1}{k-1})^t \leq ke^{-t/k}$.

Now let T denote the random variable that counts the number of 0-answers needed to determine the secret code. By a union bound over all n positions, we have $\Pr[T \geq t] \leq nke^{-t/k} = \Theta(n^2) \cdot e^{-\Theta(t/n)}$. It follows by routine calculations that $\mathbb{E}[T] = O(n \log n)$ and $\Pr[T \geq Cn \log n] = o(1)$ for C large enough. As a random query returns a value of 0 with constant probability, the same bounds also hold for the total number of queries needed. □

We now consider deterministic nonadaptive strategies to identify the secret code. Chvátal [1983] proved that the bound given in Theorem 4.1 is tight if $k \leq n^{1-\varepsilon}$, $\varepsilon > 0$ a constant. Here we extend his argument to every $k \leq n$. It essentially shows that a set of $O(\frac{n \log k}{\max\{\log(n/k),1\}})$ random guesses with high probability identifies every secret code. Our proof is based on the probabilistic method and is thus nonconstructive. It remains an open question to find an explicit nonadaptive polynomial-time strategy that achieves this bound.

THEOREM 4.3. *There exists $n_0 \in \mathbb{N}$ and a constant $C > 0$ such that for every $n \geq n_0$ and $k \leq n$ there exists a deterministic nonadaptive strategy for black-peg Mastermind with n positions and k colors that uses at most $C\frac{n \log k}{\max\{\log(n/k),1\}}$ queries.*

PROOF. The idea is to use a probabilistic method type of argument—for instance, we show that for an appropriately chosen constant $C > 0$ and n large enough, a set of $N = C\frac{n \log k}{\max\{\log(n/k),1\}}$ random guesses with positive probability identifies every possible secret code. (In fact, we will show that such a set of queries has this property with high probability.)

If a set $X = \{x^{(i)} \mid i \in N\}$ of queries distinguishes any two possible secret codes z, z', then there must exist for each such pair $z \neq z'$ a query $x \in X$ with $\mathrm{eq}(z, x) \neq \mathrm{eq}(z', x)$. In particular, we must have $|\{i \in I(z, z') : x_i = z_i\}| \neq |\{i \in I(z, z') : x_i = z'_i\}|$ for $I(z, z') := \{i \in [n] : z_i \neq z'_i\}$. Based on this observation, we define (similar to Chvátal [1983]) a *difference pattern* to be a set of indices $I \subseteq [n]$ together with two lists of colors $(c_i)_{i\in I}, (c'_i)_{i\in I}$ such that $c_i \neq c'_i$ for every $i \in I$. For every two distinct secret codes $z, z' \in [k]^n$, we define the difference pattern corresponding to z and z' to be the set $I(z, z') := \{i \in [n] : z_i \neq z'_i\}$ together with the lists $(z_i)_{i\in I}$ and $(z'_i)_{i\in I}$. We say that a query $x \in [k]^n$ *splits* a difference pattern given by I, $(c_i)_{i\in I}$, and $(c'_i)_{i\in I}$ if

$$|\{i \in I : x_i = c_i\}| \neq |\{i \in I : x_i = c'_i\}|.$$

It is now easy to see that if a set of N queries has the property that every possible difference pattern is split by at least one query from that set, then these N queries together with the answers deterministically identify Codebreaker's secret code.

In the following, we show that a set of $N = C\frac{n \log k}{\max\{\log(n/k),1\}}$ random queries with probability at least $1 - 1/n$ has the property that it splits every difference pattern.

The *size* of a difference pattern I, $(c_i)_{i\in I}$, $(c'_i)_{i\in I}$ is the cardinality of I. Note that for fixed k, the probability that a particular difference pattern is not split by a randomly chosen query only depends on its size. Let $p(d, k)$ denote this probability for a difference pattern of size d. The probability that there exists a difference pattern that is not split by any of the N random queries is at most

$$\sum_{d=1}^{n} \binom{n}{d} (k(k-1))^d (p(d, k))^N.$$

To show that this probability is at most $1/n$, it thus suffices to prove that for every $d \in [n]$, we have

$$\binom{n}{d}(k(k-1))^d(p(d,k))^N < n^{-2}. \tag{5}$$

We first take a closer look at $p(d,k)$. Observe that if a query x does *not* split a fixed difference pattern I, $(c_i)_{i\in I}$, $(c_i')_{i\in I}$, then x_i must agree with c_i on exactly half of the positions in $I' := \{i \in I \mid x_i \in \{c_i, c_i'\}\}$, and it must agree with c_i' on the other positions in I'. In particular, the size of I' must be even. More precisely, we have

$$\begin{aligned} p(d,k) &= \sum_{i=0}^{\lfloor d/2 \rfloor} \binom{d}{2i}\binom{2i}{i}\left(\frac{1}{k}\right)^{2i}\left(1-\frac{2}{k}\right)^{d-2i} \\ &= \sum_{i=0}^{\lfloor d/2 \rfloor} \binom{d}{2i}\left(\frac{2}{k}\right)^{2i}\left(1-\frac{2}{k}\right)^{d-2i}\binom{2i}{i}2^{-2i}. \end{aligned}$$

Note that $\binom{2i}{i}2^{-2i} \leq 1/2$ for every $i \geq 1$, and $1-x \leq e^{-x}$ for all $x \in \mathbb{R}$. Hence,

$$\begin{aligned} p(d,k) &\leq \left(1-\frac{2}{k}\right)^d + \frac{1}{2}\sum_{j=1}^{d}\binom{d}{j}\left(\frac{2}{k}\right)^j\left(1-\frac{2}{k}\right)^{d-j} \\ &= 1 - \frac{1}{2}\left(1-\left(1-\frac{2}{k}\right)^d\right) \\ &\leq \exp\left(-\frac{1}{2}(1-e^{-\frac{2d}{k}})\right). \end{aligned}$$

It follows that

$$\ln\frac{1}{p(d,k)} \geq \frac{1}{2}(1-e^{-\frac{2d}{k}}). \tag{6}$$

We now split the proof into two cases, $k \geq cn$ and $k < cn$, where c is a sufficiently small constant. (We determine c at the end of the proof.)

Case 1. $k \geq cn$. Observe that in this case $\log(n/k) \leq \log(1/c)$ and $\log k = \log n + \Theta(1)$. Hence, the bound claimed in Theorem 4.3 evaluates to $O(n\log n)$ in this case. It thus suffices to show that there exists a constant $C > 0$ such that $N = Cn\log n$ queries already identify every secret code with high probability.

We show that $n^{5d}(p(d,k))^N < 1$ for every $d \in [n]$, which clearly implies (5). In fact, we show the equivalent inequality

$$\frac{N}{5d}\ln\frac{1}{p(d,k)} > \ln n. \tag{7}$$

Using (6), we obtain

$$\frac{N}{5d}\ln\frac{1}{p(d,k)} \geq \frac{N}{10}\frac{1-e^{-\frac{2d}{k}}}{d}. \tag{8}$$

Using that $d \mapsto (1-e^{-2d/k})/d$ is a decreasing function in d, we can continue with

$$\frac{N}{5d}\ln\frac{1}{p(d,k)} \geq \frac{N}{10}\frac{(1-e^{-\frac{2n}{k}})}{n},$$

which is clearly larger than $\ln n$ for any $N > \frac{10}{(1-e^{-2})\log e} n \log n$. Hence, for such N we have (7), which settles this case.

Case 2. $k < cn$. In this case, we need to be more careful in our analysis, as in our claimed bound the factor $\log(n/k)$ might be large and the factor $\log k$ might be substantially smaller than $\log n$.

In what follows, we regard only the case $k \geq 3$; the case $k = 2$ has already been solved (see Erdős and Rényi [1963]).

We first consider difference patterns of size $d \leq \frac{n \log k}{\log(n/k)\log n}$. As in Case 1, we show that (7) holds for these patterns. Observe that (8) holds again in this case. Since the function $d \mapsto (1 - e^{-2d/k})/d$ is decreasing in d and since $d \leq \frac{n \log k}{\log(n/k)\log n}$, we obtain

$$\frac{N}{5d} \ln \frac{1}{p(d,k)} \geq \frac{N}{10} \frac{\left(1 - e^{-\frac{2n \log k}{k \log(n/k) \log n}}\right) \log(n/k) \log n}{n \log k}. \tag{9}$$

Next we bound the exponent $\frac{n \log k}{k \log(n/k) \log n}$ in the previous expression. Note that the derivative of $\frac{n \log k}{k \log(n/k) \log n}$ with respect to k is

$$\frac{n(\log n - \ln(2) \log k \log(n/k))}{\ln(2) k^2 \log n \log^2(n/k)}. \tag{10}$$

We now show that this expression is less than 0 for $3 \leq k \leq n/4$. Indeed, observe that by setting $g(k) = \ln(2) \log k \log(n/k)$, we have for n large enough that $g(3) = \ln(2)\log(3)\log(n/3) > 1.09 \log(n) - 3.3 > \log n$ and $g(n/4) = 2\ln(2)\log(n/4) > \log n$. Moreover, observe that

$$g'(k) = \frac{\log n - \log(k^2)}{k}.$$

From this, one easily sees that the function g has a local maximum at $k = \sqrt{n}$ as its only extremal point in the interval $3 \leq k \leq n/4$. Hence, $g(k) > \log n$ for every $3 \leq k \leq n/4$, and thus (10) is negative.

Therefore, $\frac{n \log k}{k \log(n/k) \log n}$ is a decreasing function in k, and we have $\frac{n \log k}{k \log(n/k) \log n} \geq \frac{1}{c \log(1/c)}(1 + \frac{\log c}{\log n}) \geq 1$ for n large enough. With this, we can continue (9) with

$$\frac{N}{5d} \ln \frac{1}{p(d,k)} \geq \frac{1 - e^{-2}}{10} \frac{N \log(n/k) \log n}{n \log k},$$

which is certainly larger than $\ln n$ for any $N \geq \frac{10}{(1-e^{-2})\log e} \frac{n \log k}{\log(n/k)}$. This settles the case $k < cn$ for all $d \leq \frac{n \log k}{\log(n/k)\log n}$.

In the remainder of this proof, we consider the case $k < cn$ and $d \geq \frac{n \log k}{\log(n/k)\log n}$. For such d, we establish the inequality $2^n k^{2n} (p(d,k))^N < n^{-2}$, which clearly implies (5). As done previously, we actually show the equivalent inequality

$$N \log \frac{1}{p(d,k)} > 2n \log k + n + 2 \log n. \tag{11}$$

First observe that $\binom{2i}{i}2^{-2i} \leq 1/\sqrt{i}$ for every $i \geq 1$. We denote by $\mathcal{B}in(n, p)$ a binomially distributed random variable with parameters n and p. With this, we obtain

$$\begin{aligned} p(d,k) &\leq \sum_{j=0}^{\lfloor d/k \rfloor} \binom{d}{j}\left(\frac{2}{k}\right)^j\left(1-\frac{2}{k}\right)^{d-j} + \left(\frac{d}{k}\right)^{-1/2} \sum_{j=\lfloor d/k \rfloor+1}^{n} \binom{d}{j}\left(\frac{2}{k}\right)^j\left(1-\frac{2}{k}\right)^{d-j} \\ &= \Pr\left[\mathcal{B}in\left(d, \frac{2}{k}\right) \leq \frac{d}{k}\right] + \left(\frac{d}{k}\right)^{-1/2} \sum_{j=\lfloor d/k \rfloor+1}^{n} \binom{d}{j}\left(\frac{2}{k}\right)^j\left(1-\frac{2}{k}\right)^{d-j}. \end{aligned}$$

Using the Chernoff bound $\Pr[\mathcal{B}in(n, p) \leq (1-\delta)np] \leq (\frac{e^{-\delta}}{(1-\delta)^{1-\delta}})^{np}$, we obtain

$$\Pr\left[\mathcal{B}in\left(d, \frac{2}{k}\right) \leq \frac{d}{k}\right] \leq \left(\frac{e^{-1/2}}{(1/2)^{1/2}}\right)^{2d/k} = \left(\frac{2}{e}\right)^{d/k}.$$

Hence, we have

$$p(d,k) \leq \left(\frac{2}{e}\right)^{d/k} + \left(\frac{d}{k}\right)^{-1/2}.$$

It is not hard to see that the function

$$f(k) = \frac{\left(\frac{2}{e}\right)^{d/k}}{\left(\frac{d}{k}\right)^{-1/2}}$$

attains its maximum at $k = 2(1-\ln 2)d$ and that $f(2(1-\ln 2)d) \leq 1$. Therefore, we have

$$p(d,k) \leq 2\left(\frac{d}{k}\right)^{-1/2} = \left(\frac{d}{4k}\right)^{-1/2}.$$

With this, we obtain

$$\begin{aligned} N \log \frac{1}{p(d,k)} &\geq \frac{N}{2}(\log d - \log k - 2) \\ &\geq \frac{N}{2}(\log n + \log\log k - \log\log(n/k) - \log\log n - \log k - 2) \\ &= \frac{N}{2}\left(\log(n/k) - \log\log(n/k) - \log\left(\frac{\log n}{\log k}\right) - 2\right) \\ &\geq \frac{N}{4}\log(n/k), \end{aligned}$$

where the last inequality follows from $\frac{1}{2}\log(n/k) - \log\log(n/k) - \log(\frac{\log n}{\log k}) - 2 \geq 0$ for every $k \leq cn$ for a sufficiently small constant $c > 0$ and n large enough. (In fact, this step imposes the most restrictive bound on c, i.e., any $c > 0$ that, for n large enough, satisfies $\frac{1}{2}\log(1/c) - \log\log(1/c) - \log(1 - \frac{\log c}{\log cn}) - 2 \geq 0$ is appropriate for our proof.) Clearly, $\frac{N}{4}\log(n/k)$ is larger than $2n\log k + n + 2\log n$ for any $N > 16\frac{n\log k}{\log(n/k)}$ and n large enough. This implies (11) and thus settles this last case. $\square$

REFERENCES

Lotte Berghman, Dries Goossens, and Roel Leus. 2009. Efficient solutions for Mastermind using genetic algorithms. *Computers and Operations Research* 36, 1880–1885.

Nader H. Bshouty. 2009. Optimal algorithms for the coin weighing problem with a spring scale. In *Proceedings of the 22nd Conference on Learning Theory (COLT'09)*.

David G. Cantor and William H. Mills. 1966. Determination of a subset from certain combinatorial properties. *Canadian Journal of Mathematics* 18, 42–48.

Zhixiang Chen, Carlos Cunha, and Steven Homer. 1996. Finding a hidden code by asking questions. In *Proceedings of the 2nd Annual International Conference on Computing and Combinatorics (COCOON'96)*. 50–55.

Vasek Chvátal. 1983. Mastermind. *Combinatorica* 3, 325–329.

Benjamin Doerr and Carola Winzen. 2014. Playing Mastermind with constant-size memory. *Theory of Computing Systems* 55, 658–684.

Stefan Droste, Thomas Jansen, and Ingo Wegener. 2006. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems* 39, 525–544.

Paul Erdős and Alfréd Rényi. 1963. On two problems of information theory. *Magyar Tudományos Akadémia Matematikai Kutató Intézet Közleményei* 8, 229–243.

Riccardo Focardi and Flaminia L. Luccio. 2010. Cracking bank PINs by playing Mastermind. In *Proceedings of the 5th International Conference on Fun with Algorithms (FUN'10)*. 202–213.

Wayne Goddard. 2003. Static Mastermind. *Journal of Combinatorial Mathematics and Combinatorial Computing* 47, 225–236.

Wayne Goddard. 2004. Mastermind revisited. *Journal of Combinatorial Mathematics and Combinatorial Computing* 51, 215–220.

Michael T. Goodrich. 2009a. The Mastermind attack on genomic data. In *Proceedings of the 2009 30th IEEE Symposium on Security and Privacy (SP'09)*. IEEE, Los Alamitos, CA, 204–218.

Michael T. Goodrich. 2009b. On the algorithmic complexity of the Mastermind game with black-peg results. *Information Processing Letters* 109, 675–678.

Vladimir Grebinski and Gregory Kucherov. 2000. Optimal reconstruction of graphs under the additive model. *Algorithmica* 28, 104–124. DOI:http://dx.doi.org/10.1007/s004530010033

Juan J. Merelo Guervós, Carlos Cotta, and Antonio Mora García. 2011a. Improving and scaling evolutionary approaches to the Mastermind problem. In *Proceedings of the Conference on Applications of Evolutionary Computation (EvoApplications'11)*. 103–112.

Juan J. Merelo Guervós, Antonio Miguel Mora, and Carlos Cotta. 2011b. Optimizing worst-case scenario in evolutionary solutions to the Mastermind puzzle. In *Proceedings of the IEEE Congress on Evolutionary Computation (CEC'11)*. IEEE, 2669–2676.

Philippe Jacquet and Wojciech Szpankowski. 1999. Entropy computations via analytic depoissonization. *IEEE Transactions on Information Theory* 45, 1072–1081.

Gerold Jäger and Marcin Peczarski. 2009. The number of pessimistic guesses in generalized Mastermind. *Information Processing Letters* 109, 635–641.

Gerold Jäger and Marcin Peczarski. 2011. The number of pessimistic guesses in generalized black-peg Mastermind. *Information Processing Letters* 111, 933–940.

Tom Kalisker and Doug Camens. 2003. Solving Mastermind using genetic algorithms. In *Proceedings of the Conference on Genetic and Evolutionary Computation (GECCO'03)*. ACM, New York, NY, 1590–1591.

Donald E. Knuth. 1977. The computer as a master mind. *Journal of Recreational Mathematics* 9, 1–5.

Kenji Koyama and Tony W. Lai. 1993. An optimal mastermind strategy. *Journal of Recreational Mathematics* 25, 251–256.

Bernt Lindström. 1964. On a combinatory detection problem I. *Publications of the Mathematical Institute of the Hungarian Academy of Science* 9, 195–207.

Bernt Lindström. 1965. On a combinatorial problem in number theory. *Canadian Mathematical Bulletin* 8, 477–490.

Robin A. Moser. 2009. A constructive proof of the Lovász local lemma. In *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC'09)*. ACM, New York, NY, 343–350. DOI:http://dx.doi.org/10.1145/1536414.1536462

A. Pelc. 2002. Searching games with errors—fifty years of coping with liars. *Theoretical Computer Science* 270, 71–109.

J. Spencer. 1994. Randomization, derandomization and antirandomization: Three games. *Theoretical Computer Science* 131, 415–429.

Jeff Stuckman and Guo-Qiang Zhang. 2006. Mastermind is NP-complete. *INFOCOMP Journal of Computer Science* 5, 25–28.

Alexandre Temporel and Tim Kovacs. 2003. A heuristic hill climbing algorithm for Mastermind. In *Proceedings of the 2003 UK Workshop on Computational Intelligence (UKCI'03)*. 189–196.

Giovanni Viglietta. 2012. Hardness of Mastermind. In *Proceedings of the 6th International Conference on Fun with Algorithms (FUN'12)*. 368–378.

Received January 2013; revised July 2016; accepted August 2016