Nash Equilibrium in Mastermind

François Bonnet(✉) and Simon Viennot(✉)

Graduate School of Advanced Science and Technology, JAIST, Nomi, Japan
{f-bonnet,sviennot}@jaist.ac.jp

Abstract. Mastermind is a famous two-player deduction game. A Codemaker chooses a secret code and a Codebreaker tries to guess this secret code in as few guesses as possible, with feedback information after each guess. Many existing works have computed optimal worst-case and average-case strategies of the Codebreaker, assuming that the Codemaker chooses the secret code uniformly at random. However, the Codemaker can freely choose any distribution probability on the secret codes. An optimal strategy in this more general setting is known as a Nash Equilibrium. In this research, we compute such a Nash Equilibrium for all instances of Mastermind up to the most classical instance of 4 pegs and 6 colors, showing that the uniform distribution is not always the best choice for the Codemaker. We also show the direct relation between Nash Equilibrium computations and computations of worst-case and average-case strategies.

1 Introduction

1.1 A Simple Deduction Game

Before studying the famous game of Mastermind, let us start by describing a much simpler deduction game to illustrate some notions. We consider the two-player game *Guess-A-Number*, played by Alice and Bob. Initially, Alice chooses a *secret number* between 1 and N, where N is a parameter of the game (known by both players). Bob has to discover Alice's secret number. To find this number, Bob makes successive *guesses*; each guess being a number. The game ends when Bob correctly guesses her number. After each guess, Alice gives a *feedback* informing Bob whether (a) his guess is the secret number, or (b) his guess is lower than the secret number, or (c) his guess is higher. Based on that feedback, Bob eventually finds Alice's secret number. In this game, Bob's goal is to win using as few guesses as possible. Note that Alice does not have any choice except at the beginning, when she chooses her number.

Example. Let consider the smallest non-trivial instance of this game, with $N = 3$, and let us analyze what are the possible strategies of the players:

- Alice chooses a number from the set $\{1, 2, 3\}$.
- Bob is clever; he knows that his first guess should be the number 2. It is the only first guess that guarantees him to discover Alice's number in at most two steps *in the worst case*. Indeed, (a) if Alice chose the number 2, Bob wins

A. Plaat et al. (Eds.): CG 2016, LNCS 10068, pp. 115–128, 2016.
DOI: 10.1007/978-3-319-50935-8_11

in a single guess; while (b) if Alice chose 1 or 3, he wins using two guesses. Assuming that Alice chooses her number *uniformly* at random, Bob's strategy requires $\frac{2+1+2}{3} \simeq 1.67$ steps in *average*.
- Alice is also clever; she is expecting Bob to first guess the number 2. To counter him, she decides to avoid choosing number 2 as her secret number; she selects now her secret number only between 1 and 3 with equal probability. This change in her strategy increases the average number of guesses of Bob; it becomes $\frac{2+0+2}{2} = 2$.
- Bob is even more clever. He knows that Alice never chooses the number 2 as her secret number. Then he updates his strategy too. Instead of guessing first the number 2, he only needs to guess 1 and 3 successively. His average number of guesses reduces to $\frac{1+0+2}{2} = 1.5$.
- Alice is ...

Such reasoning could continue forever and Alice's and Bob's strategies will never converge. Fortunately, many years ago, Nash introduced the notion of Nash Equilibrium (NE) [11]. There exist some pairs of players' strategies for which none of the players benefit in changing their strategy. In the previously-described game, when both players are playing optimal strategies (in the NE sense), Bob discovers the number in an average of $\frac{9}{5} = 1.8$ guesses. Alice has to choose number 1 with probability $\frac{2}{5}$, number 2 with probability $\frac{1}{5}$, and number 3 with probability $\frac{2}{5}$.[1]

Mastermind is a more complex game, but a similar analysis can be done, as we will see in this paper.

1.2 The Game of Mastermind

Mastermind is a two-player game in which Alice (*aka.* the Codemaker) chooses a *secret code* and Bob (*aka.* the Codebreaker) has to discover this secret code. The code is a sequence of n pegs, each of them taken from a set of k colors. The original game, invented and sold in the 1970s, proposed a code of length $n = 4$ and $k = 6$ colors, leading to $6^4 = 1\,296$ possible secret codes. Later, to increase the level of difficulty, a *Super Mastermind* version was commercialized with $n = 5$ pegs and $k = 8$ colors.

After Alice has chosen her secret code, Bob makes successive *guesses*; each guess being a sequence of n pegs chosen among k colors (*i.e.* a guess corresponds to one of the possible codes). After each guess, Alice gives a feedback (*aka. grade*) informing Bob about the quality of his guess. The grade is a pair of two integers[2] (b, w), where b indicates the number of correct pegs in the guess with respect to the secret code, and w indicates the number of pegs with correct colors but incorrect positions (see Sect. 2 for a more formal definition).

[1] The straightforward proof is left to the reader.

[2] We use (b, w) since the original game use *b*lack and *w*hite pins to display the grade.

Example. Consider the game instance with $n = 5$ and $k = 8$, where colors are represented by numbers 1 to 8. Alice chooses the secret code 72321 and Bob proposes the guess 23523. In Bob's guess, only one single peg is correct (good position, good color); the one at position 4 with color 2. There are also two pegs with a correct color but an incorrect position: the peg with color 2 at position 1 and the peg with color 3 at position 2. Consequently, the grade is $(b, w) = (1, 2)$. Note that since the secret code contains a single peg with color 3; only one peg with color 3 from Bob's guess is counted in the number w.

1.3 Related Work

Due to its mathematical nature, the game has been widely studied. In 1976, Knuth analyzed the original game $(n, k) = (4, 6)$ and proved that any code can be found using at most 5 guesses [9], and one cannot do better; there is no strategy solving all codes in 4 guesses. After this optimal *worst case* result, researchers started naturally to look for an optimal *average case* result. In 1993, Koyama and Lai finally proved that the best strategy finds the code in $\frac{5625}{1296} \simeq 4.34$ guesses in average (assuming that the secret code is chosen *uniformly at random* among the 1296 possible codes) [10]. Similar worst and average cases have later been computed for larger instances; latest results up to $(4, 7)$ are given by Ville [13].

A general average case solution for two pegs and arbitrary number of colors is given by Goddard [5] and Chen and Lin [1]. General worst case solutions for three pegs (and arbitrary number of colors) or two colors (and arbitrary number of pegs) have been proposed by Jager [7]. Asymptotic bounds for large number of pegs were first proposed in 1983 by Chvátal [2] and recently improved by Doerr *et al.* [3]. Related works also include variants of the game with only black pegs [6,8] or in a static version [2,4] where the goal is to determine the code by asking simultaneously many guesses (without waiting for the feedback).

1.4 Motivation and Contributions

All related work mentioned in the previous paragraph deals with Codebreaker strategies. This is quite natural, since this is what makes the game interesting for human players. However Mastermind, similarly to the Guess-A-Number game, is in fact a two-player game. One should also investigate the strategies of the Codemaker. She plays only a single move in the game (choosing the secret code) but this choice may have an impact on the strategies of the Codebreaker.

To the best of our knowledge, only one paper has been published on this specific problem. In 1982, Pearson proposed an analysis of *two person, zero sum games* and illustrated his theory using a small instance of Mastermind [12]. When playing with $n = 2$ pegs and $k = 3$ colors, the codebreaker should choose any *unicolor*[3] code with probability $\frac{1}{6}$ and any *bicolor* code with probability $\frac{1}{12}$. This distribution of secret codes guarantees the highest possible average case for

[3] A unicolor (resp. bicolor) code is a code with both pegs with same (resp. different) color. There are 3 unicolor codes and 6 bicolor codes. Note that $3 \times \frac{1}{6} + 6 \times \frac{1}{12} = 1$.

the Codebreaker, namely $\frac{29}{12} \simeq 2.42$ guesses (compared to $\frac{21}{9} \simeq 2.33$ guesses if the Codemaker plays uniformly at random).

In 1995, Wiener posted a message on `sci.math` newsgroup [14] stating that he has "done a full game-theoretic analysis of the 4-peg, 6-colour version of mastermind [...]". He concluded that an optimal Codemaker must not play any unicolor code, but should play any other code uniformly at random, *i.e.* with probability $\frac{1}{1290}$. Such distribution of code leads to an average case of $\frac{5600}{1290} \simeq 4.34$ guesses for an optimal Codebreaker.

However, as far as we know, Wiener did not release any publication of this result, except this short newsgroup message. The result has never been computed independently and there is also no existing description of the computation algorithms. In this research, our goal is then to compute such optimal Codemaker strategy (more precisely, to compute the Nash Equilibrium) for different instances of Mastermind. We confirm the announced result of Wiener and we also show that smaller instances of Mastermind exhibit interesting Nash Equilibria.

1.5 Outline

The remaining of the paper is organized as follows. Section 2 formalizes the game and summarizes some of the known results. Section 3 explains how we computed Nash Equilibria. Section 4 presents our results emphasizing some unexpected cases. Section 5 finally concludes the paper.

2 Definitions and Notations

2.1 Rules of the Game

As described in the Introduction, Mastermind is a two-player game where Alice (the Codemaker) chooses a secret code that Bob (the Codebreaker) has to discover. The game is parametrized by a pair of integers (n, k) where n denotes the number of pegs (*i.e.* the length of a code) and k denotes the number of colors (*i.e.* the cardinality of the alphabet). In this paper, we use integers to represent colors; an alphabet of k colors is represented by the set $\{0, 1, 2, \ldots, k-1\}$. A code C is therefore a sequence of n integers; $C = (C_1, C_2, C_3, \ldots, C_n)$. Since all proposed examples contain at most ten colors, we simplify notations as follows: code $(1, 0, 3, 2, 0)$ becomes 10320.

To discover the secret code, the Codebreaker makes successive guesses, each guess being a code. The game ends when he successfully guesses the correct code. The Codemaker grades each submitted guess with respect to her secret code. The grading function $g_{n,k}$ is a symmetric function taking two codes as inputs and returning a pair of integers (b, w) as output. For an instance (n, k) of the game of Mastermind, a secret code S and a guess G, $g_{n,k}(S, G) = (b, w)$ with:

$$b = \sum_{i=1}^{n} \delta(S_i, G_i) \qquad \text{and} \qquad w = \max_{\tilde{G} \in Perm(G)} \left(\sum_{i=1}^{n} \delta(S_i, \tilde{G}_i) \right) - b,$$

where δ denotes the Kronecker symbol and $Perm(G)$ denotes the set of all permutations of G (formalization inspired from [2]).

Combinatorial observations. Given an instance (n, k) of the game, there are k^n possible codes. Both integers b and w output by the grading function belong to the set $\{0, \ldots, n\}$ and they satisfy the inequality $b + w \leq n$. There is clearly no pair of codes with relative grade equal to $(n-1, 1)$. Thus, for a given instance (n, k) of Mastermind with $k > 2$, there are $\frac{n(n+3)}{2}$ possible grades.[4]

2.2 Strategies in the Game

In the following, we assume a fixed instance (n, k) of the game. Formally, all functions defined below (such as wc) are parametrized with n and k (and should be denoted $wc_{n,k}$). For clarity, we do not subscript these functions.

Codemaker's Strategies. The only move of the Codemaker is to choose a secret code. A pure strategy is to choose a given code. Usually we consider only mixed strategies where the Codemaker plays according to a distribution of probability over the set of all possible codes. One particular strategy is the Uniform strategy where she chooses a secret code uniformly at random among the k^n codes.

Codebreaker's Strategies. The Codebreaker has to find the secret code. In the game of Mastermind, pure strategies of the Codebreaker can easily be represented by trees where nodes are guesses and edges are grades. To play a given strategy/tree, the Codebreaker starts with the root node as initial guess, and then follows the edge corresponding to the grade received from the Codemaker, until it reaches a leaf meaning that the secret code has been found.

The Codebreaker wants to win *as quickly as possible*. Formally, this notion can be interpreted in (at least) three ways. Given a (pure or mixed) strategy S of the Codebreaker, we define the three following values:

1. **Worst case:** $wc(S)$ denotes the smallest integer such that S is guaranteed to discover any secret code in at most $wc(S)$ guesses. A strategy S is said to be wc-optimal if there is no strategy S' with $wc(S') < wc(S)$.
   For a pure strategy S, $wc(S)$ equals the height of the corresponding tree.
2. **Average case:** $avg(S)$ denotes the average number of guesses required to discover any code, assuming that the Codemaker plays her Uniform strategy. A strategy is said to be avg-optimal if there is no strategy S' with $avg(S') < avg(S)$.
   For a pure strategy S, $avg(S)$ equals the average depth of all leaves.

[4] This formula is incorrect when $k = 1$ since there exists a single code, hence a single possible grade $(n, 0)$. For $k = 2$ colors, there are also some unreachable grades such as $(b, w) = (0, n)$ when n is odd.

3. **Weakest case:** $weak(S)$ denotes the largest average number of guesses required when playing against all possible Codemaker strategies. Said differently, it corresponds to the average number of guesses required when the adversary is playing the strongest possible strategy that counters the Codebreaker's strategy.
   For a pure strategy S, $weak(S) = wc(S)$ since the Codemaker can choose to play (one of) the pure strategy corresponding to (one of) the worst case. Thus this notion of $weak(S)$ is really interesting only for mixed strategies.

When considering an instance (n, k) of Mastermind, we define $\mathrm{WC}(n, k)$ to denote the value of some wc-optimal strategy. Similarly, we use the notation $\mathrm{AVG}(n, k)$ for avg-optimal strategy:

$$\mathrm{WC}(n,k) = \min_{S \in \mathbb{S}_B} (wc(S)) \qquad \text{and} \qquad \mathrm{AVG}(n,k) = \min_{S \in \mathbb{S}_B} (avg(S)),$$

where $\mathbb{S}_B$ denotes the set of all strategies of the Codebreaker, including mixed strategies.[5] As already mentioned in Sect. 1.3, existing related work deals with computing these values $\mathrm{WC}(n, k)$ and $\mathrm{AVG}(n, k)$.

Nash Equilibrium. The definitions of WC and AVG are based on the notions of wc-optimality and avg-optimality respectively. Similarly one can use the third notion of weakest case to define a last optimality criteria, corresponding to the Nash Equilibrium $\mathrm{NE}(n, k)$:

$$\mathrm{NE}(n,k) = \min_{S \in \mathbb{S}_B} (weak(S)).$$

2.3 Example with $(n, k) = (3, 2)$

To illustrate the previous definitions, let us consider a simple instance of Mastermind with 3 pegs and 2 colors. There are $2^3 = 8$ codes; 000, 001, 010, 011, 100, 101, 110, 111. Figure 1 depicts a possible strategy S for the Codebreaker. Note that there is no edge labeled $(0, 0)$, $(0, 1)$, $(0, 3)$, or $(1, 1)$ starting from the root since no secret code can lead to such grade when compared with guess 001.

Using this strategy, in the worst case, the Codebreaker wins in four guesses (to find the secret code 101), hence $wc(S) = 4$. Globally, one code is found in 1 guess, four codes in 2 guesses, two codes in 3 guesses, and one code in 4 guesses, hence $avg(S) = \frac{1\times1+4\times2+2\times3+1\times4}{8} = \frac{19}{8} \simeq 2.38$. This strategy S is neither wc-optimal nor avg-optimal. Indeed there exists better strategies for both metrics; $\mathrm{WC}(3, 2) = 3$ and $\mathrm{AVG}(3, 2) = \frac{18}{8} = 2.25$ (finding them is left to the reader).

[5] To compute WC and AVG, it is sufficient to consider only pure strategies. However for NE, it is required to consider also mixed strategies.

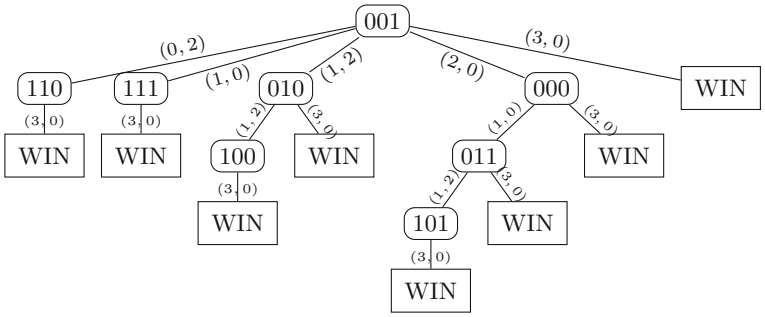

Fig. 1. A Codebreaker's strategy S for the instance $(3, 2)$ of Mastermind

Since S is a pure strategy, as already mentioned earlier, $weak(S) = wc(S) = 4$. Indeed, knowing that the Codebreaker is playing S, the Codemaker may decide to always choose the secret code 101.

2.4 Known Results

Table 1 summarizes known values of AVG for Mastermind. As far as we know, blank entries (*e.g.* AVG$(5, 5)$) are still unknown; indeed even relatively small instances are hard to compute due to combinatorial explosion. This table is not exhaustive; Ville lists additional values, such as AVG$(4, 7)$ [13].

Table 1. Some known values of AVG(n, k)

		Number of colors k				
		2	3	4	5	6
Nb. of pegs n	2	$8/4 = 2$	$21/9 \approx 2.33$	$45/16 \approx 2.81$	$81/25 = 3.24$	$132/36 \approx 3.67$
	3	$18/8 = 2.25$	$73/27 \approx 2.70$	$206/64 \approx 3.22$	$451/125 \approx 3.61$	$854/216 \approx 3.95$
	4	$44/16 = 2.75$	$246/81 \approx 3.04$	$905/256 \approx 3.54$	$2463/625 \approx 3.94$	$5625/1296 \approx 4.34$
	5	$97/32 \approx 3.03$	$816/243 \approx 3.36$	$3954/1024 \approx 3.86$		

3 How to Compute Optimal Values

This section explains how to compute WC, AVG, and NE values.

3.1 Basic Approach

The naive approach involves generating all pure strategies for the Codebreaker[6] and analyzing them to obtain the desired values. Once the list of all strategies

[6] Pure strategies of the Codemaker are trivially computed. There are exactly k^n pure strategies, one for each possible code.

has been computed, one can evaluate each of them by playing them against all pure strategies of the Codemaker. We denote with $res(S)$ the *result* of a strategy S; $res(S)$ is an k^n-tuple where the i^{th} element indicates the number of guesses required to find the secret code when the Codemaker has chosen the i^{th} code (assuming, wlog. a natural lexicographic ordering of codes).

Observing that $wc(S) = \max(res(S))$ and $avg(S) = \frac{\sum res(S)}{k^n}$, one can compute WC and AVG from the set of strategies. This approach becomes quickly impractical due to combinatorial explosion; the number of strategies becomes intractable. Out of curiosity, we computed the total number of strategies for the Codebreaker. They appear as the first number of the fourth column of Table 2. The second number of the same fourth column indicates the number of unique strategy results. Indeed, many strategies lead to the same result. Considering only unique results already reduces greatly the order of magnitude.

Example (continued). Considering the strategy S of Fig. 1, one can check that $res(S) = (2, 1, 2, 3, 3, 4, 2, 2)$.

Table 2. Some interesting numbers about the combinatorial explosion. #strategies represents the total number of strategies and the total number of unique results. #results represents, with respect to equivalence classes, (i) the total number of unique results, (ii) the total number after eliminating results dominated by other results, (iii) an approximate number of non-dominated results (domination by linear combination).

(n, k)	#codes	#grades	#strategies	#classes	#results (wrt classes)
$(2, 2)$	4	5	8 – 8	2	2 – 2 – 2
$(2, 3)$	9	5	26 760 – 1 278	2	33 – 3 – 3
$(2, 4)$	16	5	2.08×10^{11} – 6 043 176	2	188 – 6 – 3
$(2, 5)$	25	5	9.91×10^{21} –	2	557 – 8 – 5
$(2, 6)$	36	5	9.29×10^{36} –	2	1 377 – 11 – 5
$(3, 2)$	8	9	1 776 – 648	2	16 – 3 – 3
$(3, 3)$	27	9	2.47×10^{23} –	3	2 489 – 17 – ∼12
$(3, 4)$	64	9	1.47×10^{79} –	3	124 852 – 112 – ∼24
$(3, 5)$	125	9	6.72×10^{190} –	3	1 201 354 – 286 – ∼69
$(3, 6)$	216	9	–	3	6 793 325 – 619 – ∼123
$(4, 2)$	16	14	2.29×10^{11} – 14 578 420	3	230 – 4 – 4
$(4, 3)$	81	14	4.13×10^{107} –	4	2 669 925 – 509 – ∼143
$(4, 4)$	256	14	–	5	– 107 274 – ∼12 430
$(4, 5)$	625	14	–	5	– 4 650 433 – ∼200 604
$(4, 6)$	1296	14	–	5	– – ∼899 057

Table 3. Number of codes ($= k^n$) and number of equivalence classes

		Number of colors k				
		2	3	4	5	6
Nb. of pegs n	2	$4 \Rightarrow 2$	$9 \Rightarrow 2$	$16 \Rightarrow 2$	$25 \Rightarrow 2$	$36 \Rightarrow 2$
	3	$8 \Rightarrow 2$	$27 \Rightarrow 3$	$64 \Rightarrow 3$	$125 \Rightarrow 3$	$216 \Rightarrow 3$
	4	$16 \Rightarrow 3$	$81 \Rightarrow 4$	$256 \Rightarrow 5$	$625 \Rightarrow 5$	$1\,296 \Rightarrow 5$
	5	$32 \Rightarrow 3$	$243 \Rightarrow 5$	$1\,024 \Rightarrow 6$	$3\,125 \Rightarrow 7$	$7\,776 \Rightarrow 7$

3.2 Equivalence Classes

The combinatorial explosion can be greatly limited by noting that from a theoretical point of view, playing a blue, red, or green peg does not make any difference. Hence any permutation of colors in a strategy S of the Codebreaker has no effect on the values of $wc(S)$, $avg(S)$, and $weak(S)$. Similarly any permutation of the pegs has also no effect. The same reasoning applies to the strategies of the Codemaker. Instead of evaluating Codebreaker's strategies against all possible codes (*i.e.* all pure Codemaker's strategies), one can "simply" evaluate strategies with respect to equivalence classes. Given a strategy S, instead of computing a k^n-tuple result, one can study a much smaller tuple whose cardinality equals the number of equivalence classes (see Table 3). $res_{eq}(S)$ is computed from $res(S)$ by summing elements corresponding to codes belonging to the same class.

Example (continued). Based on the example of Sect. 2.3 with $(n, k) = (3, 2)$, the 8 codes can be grouped in 2 equivalence classes: the class of unicolor codes $\{000, 111\}$ and the class of bicolor codes $\{001, 010, 011, 100, 101, 110\}$. Based on these classes, the new evaluation of the strategy S of Fig. 1 becomes $res_{eq}(S) = (2 + 2, 1 + 2 + 3 + 3 + 4 + 2) = (4, 15)$.

3.3 Recursive Computation with Pruning

There is in fact no need to list all possible strategies of the Codebreaker for computing the values that we are interested in. The strategies can be explored recursively by exploring alternately all possible guesses of the Codebreaker after some grade answer of the Codemaker (a grade node), and then exploring all possible grades after some guess (a guess node). Figure 2 gives an example of exploration tree for the instance $(3, 3)$.[7]

The main point of this recursive exploration is that instead of listing the strategies, we only compute the cost of grade nodes and guess nodes, which is equivalent to prune the under-optimal strategies progressively. This recursive exploration is classical for computations of worst-case and average-case, and

[7] Figures 2, 3, 4 and 5 appear only in Appendix A.

it can also be used for Nash Equilibrium. The only difference is the kind of information (cost) that will be propagated upward.

3.4 Computing WC and AVG Values

In the case of the worst-case computation, the cost of a strategy corresponds to the maximal number of guesses needed by the strategy. The cost of a grade node is the cost of the best guess choice for the Codebreaker, hence the minimal cost over the guesses. The cost of a guess node is the cost of the worst possible grade, hence the maximal cost over the grades. WC computations are similar to a mini-max algorithm.

In the case of the average-case computation, the cost of a strategy is the total number of guesses. The cost of a grade node is still the minimal cost over the guesses, but the cost of a guess node is now the sum of cost over the grades. AVG computations are similar to a mini-max algorithm but with a sum operation instead of a max operation.

3.5 Computing NE Values

The algorithm that we used to compute the Nash-Equilibrium is quite similar. The first difference is that the cost of a strategy is now represented not by a single number but by a list of numbers, *i.e.* the number of guesses needed for each equivalence class. In the case of Sect. 2.3 example, it is a couple of numbers. In the case of the usual (4,6) game, it is a 5-tuple. Also, for each grade and guess node, we cannot retain the cost of a unique optimal strategy. We need to retain the cost of all strategies that are not dominated by others.

At a grade node, the CodeBreaker can choose its next guess, so that a strategy at a grade node is any strategy for any of the guesses. The complete list of costs at a grade node is obtained by a union of the list of costs for each guess. At a guess node, a strategy of the CodeBreaker is the choice of a strategy for each of the grades. The complete list of costs is obtained by an operation that is called a Minkowski sum over the list of costs of the possible grades.[8]

The final complete list of costs that is obtained can be turned into a system of inequalities that represent the constraints on the Nash-Equilibrium. This system can be solved with classical Linear Programming methods (we used Mathematica in this research).

4 Our Results

NE Values. Table 4 summarizes the Nash Equilibrium values that we have computed. We highlighted in red the differences with AVG values of Table 1.

Strategies Achieving NE Values. Table 5 gives Alice's strategies at the equilibrium. Strategies are given with respect to classes of equivalence ordered lexicographically. For example, considering the instance $(4, 4)$, Alice should never

[8] A full description will be given in a longer version of this article.

Table 4. Computed values of NE(n, k)

Nb. of pegs n	Number of colors k = 2	3	4	5	6
2	2	29/12 ≈ 2.42	45/16 ≈ 2.81	49/15 ≈ 3.27	11/3 ≈ 3.67
3	23/10 = 2.3	73/27 ≈ 2.70	219/68 ≈ 3.22	1591/440 ≈ 3.62	619/156 ≈ 3.97
4	39/14 ≈ 2.79	67/22 ≈ 3.05	1629/460 ≈ 3.54	2463/625 ≈ 3.94	5600/1290 ≈ 4.34
5	46/15 ≈ 3.07	118/35 ≈ 3.37			

Table 5. Computed Codemaker's strategy achieving NE(n, k)

Nb. of pegs n	Number of colors k = 2	3	4	5	6
2	$\frac{1}{4}, \frac{1}{4}$	$\frac{1}{6}, \frac{1}{12}$	$\frac{1}{16}, \frac{1}{16}$	$\frac{1}{15}, \frac{1}{30}$	$\forall \alpha \in \left[\frac{1}{36}, \frac{1}{21}\right]$ $\alpha, \frac{1-6\alpha}{30}$
3	$\frac{1}{5}, \frac{1}{10}$	$\frac{1}{27}, \frac{1}{27}, \frac{1}{27}$	$\frac{1}{34}, \frac{1}{68}, \frac{1}{68}$	$\frac{1}{110}, \frac{1}{110}, \frac{3}{440}$	$\frac{1}{156}, \frac{1}{156}, \frac{1}{312}$
4	$0, \frac{1}{14}, \frac{1}{14}$	$0, \frac{1}{66}, \frac{1}{66}, \frac{1}{99}$	$0, \frac{2}{345}, \frac{1}{276}, \frac{1}{276}, \frac{1}{345}$	$\frac{1}{625}, \frac{1}{625}, \frac{1}{625}, \frac{1}{625}, \frac{1}{625}$	$0, \frac{1}{1290}, \frac{1}{1290}, \frac{1}{1290}, \frac{1}{1290}$
5	$0, \frac{1}{30}, \frac{1}{30}$	$0, \frac{3}{770}, \frac{3}{770}, \frac{1}{330}, \frac{2}{385}$			

play any of the 4 codes of the first class (class of 0000), should play each of the 48 codes of the second class (class of 0001) with probability $\frac{2}{345}$, each of the 36 codes of the third class (class of 0011) with probability $\frac{1}{276}$, each of the 144 codes of the fourth class (class of 0012) with probability $\frac{1}{276}$, each of the 24 codes of the fifth class (class of 0123) with probability $\frac{1}{345}$.[9]

Observations. These results lead to many interesting comments.

- NE values are often different from AVG value. In most cases, Alice can increase the number of guesses required by Bob. Surprisingly, for some non-trivial instances of the game, such as $(4, 5)$, she cannot improve her play; playing uniformly at random is her best option in such case.
- NE values are generally very close to AVG values. While from a theoretical point of view, both values are different, in practice it is not so bad for Alice to play uniformly at random.
- Results on optimal strategies are not trivial. No generic pattern can be deduced from the current known optimal strategies.
- Intuitively, unicolor codes are easier to solve for Bob so they should be played less frequently by Alice. This intuition is generally verified for "large" instances ($n \geq 4$), but this is not always verified, especially for small instances

[9] Fortunately, $4 \times 0 + 48 \times \frac{2}{345} + 36 \times \frac{1}{276} + 144 \times \frac{1}{276} + 24 \times \frac{1}{345} = 1$.

of the game. Sometimes, Alice has to play these unicolor codes even more frequently (*e.g.* for the game instance $(3, 4)$).
– For most of solved instances, the given optimal solution is unique.[10] Only for the game instance $(2, 6)$, Alice has an infinite number of optimal strategies.

5 Conclusion

In this research, we have computed the Nash Equilibrium of all small instances of Mastermind. We could confirm an announced but never published result about the most classical size of 4 pegs and 6 colors, which states that a uniform distribution of secret codes is not the best one for the Codemaker. She should not play unicolor secret codes. However, we found that for different numbers of pegs and colors, there is no simple rule. In the future, we plan to extend our computations to bigger sizes, and we are also working on a general result for the case of 2 pegs and an arbitrary number of colors.

Acknowledgments. This research was supported in part by JSPS KAKENHI Grant Number 26870228.

A Example for Game Instance $(n, k) = (3, 3)$

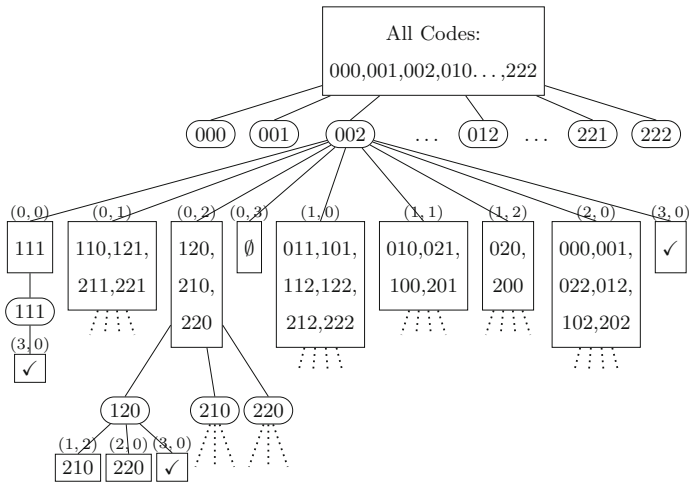

Fig. 2. Exploration tree. Squared nodes correspond to grade nodes and rounded nodes to guess nodes. Grade nodes include the list of codes still possible as the secret code. Edges from a guess node to a grade node are labeled with the corresponding grade.

[10] We could not prove yet the uniqueness for instances $(5, 3)$, $(4, 5)$, and $(4, 6)$, but it should be obtained very soon.

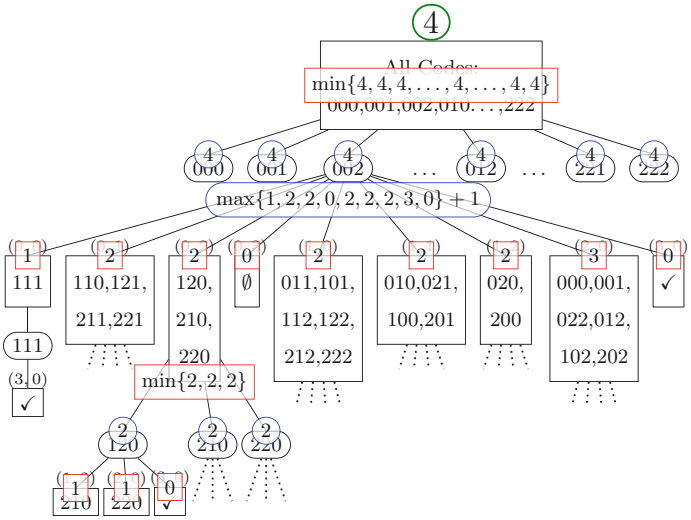

Fig. 3. Computing WC value using Min and Max operations

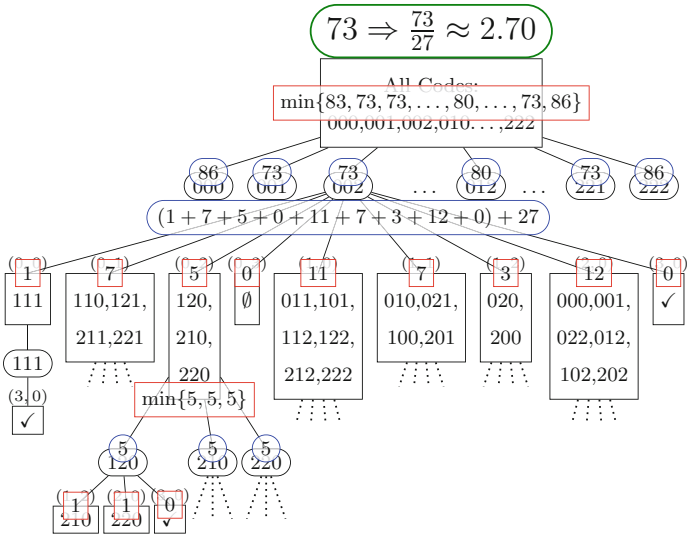

Fig. 4. Computing AVG value using Min and Sum operations

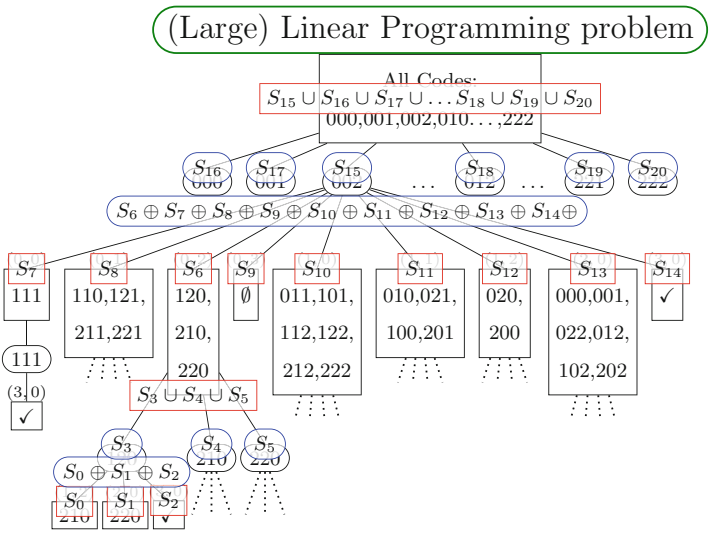

Fig. 5. Computing NE value using Union and MinkowskiSum operations

References

1. Chen, S.T., Lin, S.S.: Optimal algorithms for $2 \times n$ mastermind games-a graph-partition approach. Comput. J. **47**, 602–611 (2004)
2. Chvátal, V.: Mastermind. Combinatorica **3**, 325–329 (1983)
3. Doerr, B., Sphel, R., Thomas, H., Winzen, C.: Playing mastermind with many colors. In: Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 695–704 (2013)
4. Goddard, W.: Static mastermind. J. Comb. Math. Comb. Comput. **47**, 225–236 (2003)
5. Goddard, W.: Mastermind revisited. J. Comb. Math. Comb. Comput. **51**, 215–220 (2004)
6. Goodrich, M.T.: On the algorithmic complexity of the mastermind game with black-peg results. Inform. Process. Lett. **109**, 675–678 (2009)
7. Jäger, G., Peczarski, M.: The number of pessimistic guesses in generalized mastermind. Inform. Process. Lett. **109**, 635–641 (2009)
8. Jäger, G., Peczarski, M.: The number of pessimistic guesses in generalized black-peg mastermind. Inform. Process. Lett. **111**, 933–940 (2011)
9. Knuth, D.E.: The computer as master mind. J. Recreational Math. **9**, 1–6 (1976)
10. Koyama, K., Lai, T.: An optimal mastermind strategy. J. Recreational Math. **25**, 251–256 (1993)
11. Nash, J.F.: Non-Cooperative Games. Ph.D. thesis, Princeton University (1950)
12. Pearson, K.R.: Reducing two person, zero sum games with underlying symmetry. J. Aust. Math. Soc. (Ser. A) **33**, 152–161 (1982)
13. Ville, G.: An optimal mastermind $(4, 7)$ strategy and more results in the expected case, May 2013, eprint arXiv:1305.1010
14. Wiener, M.: Re: sci.math faq: Master mind, post in sci.math newsgroups on 29 Nov 1995 at 20: 44: 49