The number of pessimistic guesses in Generalized Mastermind

Gerold Jäger [a], Marcin Peczarski [b]

[a] *Institute of Computer Science, Martin-Luther University of Halle-Wittenberg, D-06120 Halle (Saale), Germany*
[b] *Institute of Informatics, University of Warsaw, ul. Banacha 2, PL-02-097 Warszawa, Poland*

Abstract

Mastermind is a famous two-player game, where the codemaker has to choose a secret code and the codebreaker has to guess it in as few questions as possible. The code consists of 4 pegs, each of which is one of 6 colors. In Generalized Mastermind a general number p of pegs and a general number c of colors is considered. Let $f(p, c)$ be the pessimistic number of questions for the generalization of Mastermind with an arbitrary number p of pegs and c of colors. By a computer program we compute ten new values of $f(p, c)$. Combining this program with theoretical methods, we compute all values $f(3, c)$ and a tight lower and upper bound for $f(4, c)$. For $f(p, 2)$ we give an upper bound and a lower bound. Finally, combining results for fixed p and c, we give bounds for the general case $f(p, c)$.

Key words: Combinatorial problems, Mastermind, Logic game, Computer aided proof

1. Introduction

Mastermind is a famous game invented by Mordecai Meirowitz in 1970. It is a two-player game, where the first player is called *codemaker* and the second player *codebreaker*. While the codemaker chooses a secret code, the codebreaker does not know this code and has to ask questions to guess the code. Each question is a guess for a possible secret. The codemaker has to answer each question and has to give a hint how good the codebreaker's guess is. The goal of the codebreaker is to discover the secret in as few questions as possible.

In the original game the code consists of 4 pegs, each of which is one of 6 colors, i.e., 1296 secrets are possible. The hint consists of black and white pegs, where the sum of the numbers of black and white pegs is not larger than 4. A black peg means that one peg of the codebreaker's guess is correct in position and color, and a white peg means that one peg of the guess is correct only in color. The extreme cases are 4 blacks, which means that the guess is the secret, and no whites and no blacks, which means that the colors of the secret and the guess are disjoint.

A variation of this game is *Static Mastermind*, where the codebreaker has to ask all questions at the beginning of the game and after receiving all answers has to find the secret. In 2006 the *Static Mastermind Satisfiability Problem*, has been shown to be $\mathcal{NP}$-complete [11].

In *Generalized Mastermind* a general number p of pegs and a general number c of colors is considered. The corresponding game is called $G(p, c)$. Of course, such a generalization is also possible for Static Mastermind. Much research has been done to find optimum strategies for (Static) Mastermind, where optimum strategies can mean two different points. Firstly, find a strategy such that the average number of questions over all possible secrets is minimized. For p pegs and c colors this number is denoted by $r(p, c)$. Secondly, find a strategy such that the pessimistic number of questions over all possible secrets is minimized. For p pegs and c colors this

Email addresses: jaegerg@informatik.uni-halle.de (Gerold Jäger), marpe@mimuw.edu.pl (Marcin Peczarski).

number is denoted by $f(p,c)$. Obviously, it holds $r(p,c) \leq f(p,c)$.

In 1976 Knuth presented a strategy for the original (non-static) Mastermind with $p = 4$ and $c = 6$ using a pessimistic number 5 of questions and an average number $5803/1296 \approx 4.478$ [7]. His strategy is to choose at every step the question that minimizes the maximum number of remaining possibilities. Using the information-theoretic bound explained in Section 2.2, it easily follows $f(4,6) = 5$. The exact value of $r(4,6)$ was computed by Koyama and Lai in 1993 as $5625/1296 \approx 4.34$ [8], but the strategy minimizing $r(4,6)$ requires up to 6 questions in the worst case. Values $r(p,c)$ and $f(p,c)$ for as many pairs (p,c) as possible were computed by Goddard for the non-static variant [5] and for the static variant [4]. Recently different techniques like evolutionary and genetic algorithms [1,6,10], graph partitioning [2] and a heuristic improver technique [3] were successfully applied to Mastermind.

In this paper we consider the normal (non-static) version of Generalized Mastermind and its values $f(p,c)$. Our first approach in this direction is based on a computer program. In principle, we do an exhaustive search for all possible optimum strategies. This search is realized by a recursive procedure, i.e., the search for strategies using q questions is reduced to searching for strategies using $q-1$ questions. The main reason for the efficiency of our program is the elimination of isomorphic questions. Furthermore our program helps to prove our theoretical results for 2, 3 and 4 pegs. For the case of 2 and 3 pegs we prove a formula for an arbitrary number c of colors. Whereas the result for 2 pegs has been known before, the result for 3 pegs is novel and – to the best of our knowledge – the second result for an infinite number of pegs or colors. For 4 pegs we show an upper and a lower bound. These bounds are tight up to some additive constant. Finally we consider the case of 2 colors and an arbitrary number p of pegs. We prove an upper bound and a lower bound for $f(p,2)$. Considering the values computed by our program, these bounds do not seem to be optimal. We conjecture about the exact value of $f(p,2)$. We show that proving the hypothetical exact upper bound for all odd numbers p implies it for all even ones.

The paper is organized as follows. In Section 2 we explain the basics and the main ideas of our computer program and present the new values $f(p,c)$ computed by the program. In Section 3 our results for the case of 2, 3 or 4 pegs and in Section 4 for the case of 2 colors are presented. Section 5 contains bounds for an arbitrary number of pegs and colors. We close the paper with a short summary and suggestions for future work in Section 6.

2. Computer aided methods and results

2.1. *Masets*

While playing $G(p,c)$, the possible secrets consistent with previous answers are subsets of all c^p possibilities. This motivates the definition of a subset of the possible secrets at a particular step of the game, which we call *maset*, i.e., a set during Mastermind. The size of a maset is defined as the cardinality of this subset. We denote the full maset of cardinality c^p by F. A maset of size k is called *solvable* in q questions, if the secret over the k possibilities of the maset can be found using at most q questions and the corresponding answers, i.e., the last question is answered with p black pegs. Otherwise a maset is called *unsolvable*. Define $f(M,p,c)$ as the pessimistic number of questions for solving the maset M with p pegs and c colors. Obviously we have $f(p,c) = f(F,p,c)$. We observe that a maset with size k is solvable in q questions, if $k \leq q$, i.e.,

$$f(M,p,c) \leq q, \quad \text{if } |M| \leq q. \tag{1}$$

2.2. *The information-theoretic bound*

For a given number of pegs p and questions q the *information-theoretic bound* $T(p,q)$ is an upper bound for the size of a maset which can be solved in q questions. Observe that for p pegs $(p+1)(p+2)/2$ combinations of white and black pegs exist. As the answer containing $p-1$ blacks and one white peg is impossible, we receive at most

$$A = \frac{(p+1)(p+2)}{2} - 1 = \frac{p(p+3)}{2}$$

valid answers.

$T(p,q)$ equals the maximum number of nodes in the game decision tree of height q (a one node tree has height one by this definition). In this decision tree every node represents a question, and every edge represents an answer which is not p blacks. Answers with p blacks stop the game. As at most $A-1$ further possible answers exist, every node has at most $A-1$ children. A tree has the maximum number of nodes, if every node has exactly $A-1$ children and all leaves are at the same level. We count all nodes, not only

leaves, because a game can stop in an internal node, when the answer is p blacks. Therefore

$$T(p,q) = \sum_{i=0}^{q-1} (A-1)^i. \tag{2}$$

We conclude

$$f(M,p,c) > q, \quad \text{if } |M| > T(p,q). \tag{3}$$

2.3. *The computer program*

With the purpose to compute as many values $f(p,c)$ as possible, we developed a computer program. The program is written in C++. As input parameters the program uses p, c and the pessimistic number of questions q_m and it outputs *true*, if $f(p,c) \leq q_m$ and *false* otherwise.

Inside the program we use a new recursive procedure which takes as input parameters a maset M and a number of questions q and checks, if M is solvable, i.e., if $f(M,p,c) \leq q$. In the procedure all possible questions are applied to the maset M. For each question all possible answers are considered. For each answer a new maset is produced which is a subset of the input maset M and consistent with this answer. Then the procedure is recursively applied to the new masets with $q-1$ questions. The recursion starts with the full maset F and the given number of questions q_m. The recursion stops, if M appears to be solvable because:

i) there is a question such that all masets produced as answers to this question are solvable, or
ii) the size of M is less or equal to the number of remaining questions q, i.e., (1) holds.

The recursion also stops, if M appears to be unsolvable because:

i) for every question there exists an answer which gives an unsolvable maset, or
ii) the size of M is bigger than the information-theoretic bound, i.e., (3) holds.

The masets obtained as answers to a question are sorted in decreasing order of their size. The biggest maset for each question is called *dominant*. Questions are sorted in increasing size of the dominant maset. From all masets obtained after a question the dominant one seems to be hardest to solve. A question with the smallest dominant maset seems to be the most promising question, leading at the earliest to solve a given input maset. Therefore at each recursion level we consider questions in order of increasing size of the dominant maset, starting with the most promising one. In this way, if the input maset is solvable, we reduce time needed to find the solution. On the other hand, to show that a maset is not solvable we need to consider all possible questions and for each question we need to find an answer which leads to an unsolvable maset. Therefore for each question we consider the masets in order of decreasing size, starting with the dominant one. Tests of the program confirm that the above heuristics substantially reduces the running time of the program.

2.4. *Isomorphism checking*

To further reduce the running time of our program we eliminate isomorphic questions. It is known [7] that in the original game $G(4,6)$ we need to consider in the first guess only five questions among all 1296 possible ones, namely the questions *aaaa*, *aaab*, *aabb*, *aabc* and *abcd*, where *a*, *b*, *c* and *d* are the first four of all colors. For the first guess all further questions can be omitted.

We suggest to extend this idea. Two sequences of questions are isomorphic, if there is a permutation of pegs and colors mapping one sequence to the other one. At every step of recursion the sequence of previously asked questions is extended by one question in all non-isomorphic ways. To eliminate isomorphic sequences of questions we use the *nauty* package for generating families of graphs without isomorphisms [9,12].

We represent a sequence of q questions in a game $G(p,c)$ as a colored graph. The graph contains exactly $p(q+1)+c$ vertices. The vertices are colored using $q+2$ colors denoted by $X, Y_1, \ldots, Y_q, Z$. The

Table 1
Computed values of $f(p,c)$ for $p \leq 8$ and $c \leq 10$

		c									
		1	2	3	4	5	6	7	8	9	10
p	1	1	2	3	4	5	6	7	8	9	10
	2	1	3	3	4	4	5	5	6	6	7
	3	1	3	4	4	5	5	6	**6**	**6**	**7**
	4	1	4	4	4	5	5	**6**	**6**	**7**	**7**
	5	1	4	4	5	**5**					
	6	1	5	5	**5**						
	7	1	5	**5**							
	8	1	6								

vertices $x_1, \ldots, x_p$ are colored with X and represent the pegs' positions. For $i = 1, \ldots, q$ the vertices $y_{i,1}, \ldots, y_{i,p}$ are colored with Y_i and represent the i-th question. The vertices $z_1, \ldots, z_c$ are colored with Z and represent c game colors. Every peg in a question $y_{i,j}$ is connected with vertex x_j coding its position and vertex z_k coding its color.

Two sequences of questions are isomorphic, if and only if there is a color preserving isomorphism between graphs representing these sequences. Note that the isomorphism preserves colors $X, Y_1, \ldots, Y_q, Z$, not the game colors.

2.5. *Results of computations*

The values of $f(p, c)$ computed by our program are presented in Table 1, where the new values are in bold face. The previously best values were computed by Goddard [5]. Note that in the Table of [5] wrong values for $f(2, c)$ are given (compare the theoretical result of Theorem 1). The known values were used to verify the correctness of our program. All values in Table 1 were computed in a few hours. To compute some next values we expect to need many weeks of computation.

3. Two, three and four pegs

Using our computer program, described in Section 2, we receive the following results for two, three and four pegs, and a sufficiently large number of colors.

Theorem 1 *It holds:*
a) $f(2, c) = \lfloor c/2 \rfloor + 2$ *for* $c \geq 2$,
b) $f(3, c) = \lfloor (c-1)/3 \rfloor + 4$ *for* $c \geq 5$,
c) $4 \leq f(4, c) - \lfloor c/4 \rfloor \leq 6$ *for* $c \geq 16$.

PROOF. We introduce two new games denoted by G^* and G_*. For $c \geq p^2$ let $f^*(p, c)$ be the pessimistic number of questions in a game $G^*(p, c)$, when in the p beginning questions we ask about p^2 colors, in each question p different colors. Let $f_*(p, c)$ be the pessimistic number of questions in a game $G_*(p, c)$, in which we can use one additional color, i.e., totally $c + 1$ colors, in questions. We use $f^*(p, c)$ and $f_*(p, c)$ to obtain upper and lower bounds of $f(p, c)$, respectively.

Let the number of colors be $c = kp + m$, where $k \geq p$ and $m \geq 0$. Assume that we play the game $G(p, kp + m)$ and we begin asking k questions each containing p different colors. There are at least $k - p$ empty answers which discard $(k - p)p$ colors. Hence the game $G(p, kp + m)$ is reduced to the game $G^*(p, p^2 + m)$ and we have

$$f(p, kp + m) \leq k - p + f^*(p, p^2 + m).$$

Using the computer program we have obtained for two pegs that $f^*(2, 4) = f^*(2, 5) = 4$. Hence for $k \geq 2$ and $m = 0, 1$ we have

$$f(2, 2k + m) \leq k - 2 + f^*(2, 4 + m) = k + 2$$

and for $c \geq 4$

$$f(2, c) \leq \lfloor c/2 \rfloor + 2. \tag{4}$$

Using the computer program we have obtained for three pegs that $f^*(3, 10) = f^*(3, 11) = f^*(3, 12) = 7$. Hence for $k \geq 3$ and $m = 1, 2, 3$ we have

$$f(3, 3k + m) \leq k - 3 + f^*(3, 9 + m) = k + 4$$

and for $c \geq 10$

$$f(3, c) \leq \lfloor (c-1)/3 \rfloor + 4. \tag{5}$$

Using the computer program we have obtained for four pegs and $m = 0, 1, 2, 3$ that $f^*(4, 16 + m) \leq 10$. Hence for $k \geq 4$ and $m = 0, 1, 2, 3$ we have

$$f(4, 4k + m) \leq k - 4 + f^*(4, 16 + m) = k + 6$$

and for $c \geq 16$

$$f(4, c) \leq \lfloor c/4 \rfloor + 6.$$

On the other hand, a strategy for $G(p, c)$ is also a strategy for $G_*(p, c)$. Hence $f(p, c) \geq f_*(p, c)$. We can easily transform a strategy for $G_*(p, c + 1)$ into a strategy for $G_*(p, c)$ by equating in all questions the two colors unused in a secret. Hence $f_*(p, c + 1) \geq f_*(p, c)$. If in the game $G_*(p, c)$ we ask the first question containing $m \leq p$ colors and the adversary gives us the empty answer, we are forced to play the game $G_*(p, c - m)$. Hence we have

$$\begin{aligned} f_*(p, c) &\geq 1 + \min_{1 \leq m \leq p} f_*(p, c - m) \\ &= 1 + f_*(p, c - p). \end{aligned}$$

Consequently for $c \geq c_0$ we have

$$f(p, c) \geq f_*(p, c) \geq \lfloor (c - c_0)/p \rfloor + f_*(p, c_0). \tag{6}$$

For two pegs the computer program yields $f_*(2, 2) = 3$. Hence for $c \geq 2$

$$f(2, c) \geq \lfloor (c-2)/2 \rfloor + 3 = \lfloor c/2 \rfloor + 2.$$

For three pegs the program yields $f_*(3,7) = 6$. Hence for $c \geq 7$

$$f(3,c) \geq \lfloor (c-7)/3 \rfloor + 6 = \lfloor (c-1)/3 \rfloor + 4. \qquad (7)$$

For four pegs the program yields $f_*(4,12) \geq 7$. Hence for $c \geq 12$

$$f(4,c) \geq \lfloor (c-12)/4 \rfloor + 7 = \lfloor c/4 \rfloor + 4.$$

For $p = 2 \wedge c = 2,3$ and for $p = 3 \wedge 5 \leq c \leq 9$ the inequalities (4), (5) and (7) are directly verified by our computer program (see Table 1). □

In the following we give some remarks about the proof.

i) Point a) is no new result and was proved independently in [2] and [5].
ii) The values needed in the proof of points a) and b) were computed in some seconds, those for point c) in a few hours.
iii) For the game G_* we check isomorphisms only among colors used in a secret.
iv) Transforming a strategy for $G(p, c+1)$ to a strategy for $G(p,c)$, and even proving the inequality $f(p,c) \leq f(p,c+1)$ seems to be rather difficult.
v) We conjecture that in fact $f_*(4,12) = 8$, which would improve the lower bound of $f(4,c)$ to $\lfloor c/4 \rfloor + 5$. But yet our computer program is not fast enough to prove this. The bound $f_*(4,12) \geq 7$ was obtained in 3 minutes of computation, and the computation for $f_*(4,12) \geq 8$ was not finished in one month.

4. Two colors

Considering Table 1, we conjecture that $f(p,2) = \lfloor p/2 \rfloor + 2$. Unfortunately up to now we have only been able to prove the following theorems.

Theorem 2 *For $p \in \mathbb{N}$ it holds*

$$f(p,2) \leq \lfloor 3p/4 \rfloor + 2.$$

PROOF. Let the two colors be denoted by a and b. In the following we give a strategy using at most $\lfloor 3p/4 \rfloor + 2$ questions.

Start with question

$$aaa \ldots a \qquad (Q1)$$

containing p pegs of color a. After this we know the number of a in the secret, say k.

Now we show how to solve the first four pegs of the secret in at most three further questions. Continue with question

$$bbaa\ a \ldots a. \qquad (Q2)$$

If the first two pegs are aa, we obtain $k-2$ blacks as the answer to question Q2. If the first two pegs are bb, we obtain $k+2$ blacks. If the first two pegs are ab or ba, we obtain k blacks.

Case 1. The first two pegs are aa or bb. The next two pegs can be solved by two questions as follows. Ask question

$$aaba\ a \ldots a. \qquad (Q3.1)$$

If question Q3.1 leads to $k+1$ blacks, the third peg is b. If question Q3.1 leads to $k-1$ blacks, the third peg is a. Continue with question

$$aaab\ a \ldots a. \qquad (Q4.1)$$

If question Q4.1 leads to $k+1$ blacks, the fourth peg is b. If question Q4.1 leads to $k-1$ blacks, the fourth peg is a.

Case 2. The first two pegs are ab or ba. Continue with question

$$abbb\ a \ldots a. \qquad (Q3.2)$$

Next we will consider question

$$baba\ a \ldots a. \qquad (Q4.2)$$

We have four possibilities.

i) If the answer to question Q3.2 contains $k+3$ blacks, the first four pegs are $abbb$.
ii) If the answer to question Q3.2 contains $k-3$ blacks, the first four pegs are $baaa$. In both cases we are done with the first four pegs after question Q3.2.
iii) If the answer to question Q3.2 contains $k+1$ blacks, for the first four pegs the secrets $abba$, $abab$ and $babb$ are the only ones consistent with previous answers. In this case continue with question Q4.2. Now we have three possibilities. If the answer to question Q4.2 contains $k+2$ blacks, the first four pegs are $babb$. If the answer to question Q4.2 contains k blacks, the first four pegs are $abba$. If the answer to question Q4.2 contains $k-2$ blacks, the first four pegs are $abab$.
iv) If the answer to question Q3.2 contains $k-1$ blacks, for the first four pegs the secrets $abaa$, $baab$ and $baba$ are the only ones consistent with

previous answers. Again continue with question Q4.2. Now we have again three possibilities. If the answer to question Q4.2 contains $k+2$ blacks, the first four pegs are *baba*. If the answer to question Q4.2 contains k blacks, the first four pegs are *baab*. If the answer to question Q4.2 contains $k-2$ blacks, the first four pegs are *abaa*.

Thus we have solved the first four pegs. Furthermore we know how many pegs of color a are in the last $p-4$ pegs. We can recursively repeat this step for the next four pegs by choosing in all further questions for the first four pegs the correct answer.

Now let $p = 4m + r$, where $r = 0, 1, 2, 3$. After question Q1 we ask $3m$ questions to guess the first $4m$ colors. For $r = 0$ we know all colors. For $r = 1$ we deduce the last color without any additional question, because we know the number of colors a in the secret and the number of colors a in the first $4m$ pegs. For $r = 2$ from the same reasoning we know the number of colors a in the last two pegs. Thus the cases *aa* and *bb* can be solved without any additional question and the cases *ab* and *ba* by one question. For $r = 3$ from the same reasoning we know the number of colors a in the last three pegs. Thus the cases *aaa* and *bbb* can be solved without any additional question and the remaining cases by two questions. Finally we need one question for giving the correct answer.

Overall we need $3m + 2$ questions for $r = 0$ and $r = 1$, $3m + 3$ questions for $r = 2$ and $3m + 4$ questions for $r = 3$. This leads to

$$f(p, 2) \leq \lfloor 3p/4 \rfloor + 2. \qquad \square$$

In the following we give some remarks about the proof.

Observe that starting with question Q2 the number of whites in an answer does not give any information about a secret. Let k' be the number of colors a in a question and let B, W be the number of received blacks and whites, respectively. By definition of W we have

$$\begin{aligned} W &= \min\{k, k'\} + \min\{p-k, p-k'\} - B \\ &= p - B - |k - k'|. \end{aligned} \tag{8}$$

Hence W is uniquely determined by B.

The main part of the above proof is the algorithm which after the first question $aaa \ldots a$ allows us to guess colors of 4 pegs using 3 questions. By computer search we found that similar algorithms exist for 6 and 8 pegs using 4 and 5 questions, respectively. However, they are too long to be presented here.

We can improve Theorem 2 as follows. If the remaining number of unknown peg colors is greater or equal 8 we use the procedure for 8 pegs. If it is 6 or 7 we use the procedure for 6 pegs. The color of the 7-th peg is determined, because we know the total number of pegs colored with a and b. If the number of unknown peg colors is 4 or 5 we use the procedure for 4 pegs. The color of the 5-th peg is determined as above. If the number of unknown peg colors is less than 4 we use the procedure from the second last paragraph of the above proof. This gives a bit better upper bound

$$f(p, 2) \leq \lfloor 5p/8 \rfloor + 3.$$

The next theorem gives a lower bound for the game with two colors.

Theorem 3 *For $p \geq 2$ it holds*

$$f(p, 2) \geq \frac{p}{\log_2 p}.$$

PROOF. For $p = 2, 3$ the inequality is checked directly, using Table 1. Let in the following be $p \geq 4$.

We consider the game, where we ask the first question $aaa \ldots a$. Let q be the pessimistic number of remaining questions in this game. Obviously we have $q \leq f(p, 2)$. After receiving the answer to the first question we know that the secret contains k pegs of color a and $p - k$ pegs of color b. From the information-theoretic bound we have $T(p, q) \geq |M|$, where $|M| = \binom{p}{k}$ is the size of the current maset after the first answer. The worst case happens when $\binom{p}{k}$ becomes maximum, which is for $k = p/2$ or $k = (p-1)/2$ (depending on parity of p). It is not hard to see that for $p \geq 4$ the biggest binomial coefficient $\binom{p}{k}$ is not less than $(2^p - 1)/(p - 1)$.

By (8), the number of whites in the next answers does not give any information about a secret. Hence the number of possible answers is $A = p + 1$ and by (2) we have

$$T(p, q) = \frac{p^q - 1}{p - 1}.$$

Therefore in the worst case we have

$$\frac{p^q - 1}{p - 1} \geq |M| \geq \frac{2^p - 1}{p - 1}.$$

Hence $p^q \geq 2^p$ and $f(p, 2) \geq q \geq p/\log_2 p$. $\square$

By the following theorem, to prove the tight upper bound $f(p,2) \le \lfloor p/2 \rfloor + 2$, we only have to prove the formula for p odd.

Theorem 4 *If $f(p,2) \le \lfloor p/2 \rfloor + 2$ for all p odd, then it also holds for all p even.*

PROOF. Let p be an even number. The claim for $p = 2, 4$ follows by Table 1. Suppose now $p \ge 6$. Let $f(p-1,2) \le \lfloor (p-1)/2 \rfloor + 2$ by assumption. In the following we show $f(p,2) \le \lfloor p/2 \rfloor + 2$.

Start with question

$$a \dots ab. \qquad \text{(Q1)}$$

We observe the following.

Case 1. If the answer to question Q1 contains at least two whites, the last peg is a.

Case 2. If the answer to question Q1 contains no whites and a number of blacks which is *not* $p-1$, the last peg is b.

Case 3. If the answer to question Q1 contains $p-1$ blacks and no whites, the p secrets $a \dots abb$, $a \dots abab, \dots, aba \dots ab$, $ba \dots ab$ and $a \dots a$ are the only ones consistent with the answer.

For Cases 1 and 2, the last peg is solved and we have

$$\begin{aligned} f(p,2) &\le f(p-1,2) + 1 \\ &\le \lfloor (p-1)/2 \rfloor + 3 \\ &= \lfloor p/2 \rfloor + 2. \end{aligned}$$

For Case 3 continue with question

$$\underbrace{a \dots a}_{p_1}\underbrace{b \dots b}_{p_2}a \qquad \text{(Q2)}$$

where $p_1 = \lceil (p-1)/2 \rceil$ and $p_2 = \lfloor (p-1)/2 \rfloor$. We have three possibilities.

i) If the answer to question Q2 contains $p_1 + 1$ blacks and no whites, $a \dots a$ is the only secret consistent with the answers.

ii) If the answer to question Q2 contains $p_1 + 1$ blacks and 2 whites, the secret contains b at exactly one of the pegs $p_1 + 1, \dots, p-1$. There are exactly p_2 secrets consistent with the answers. We find the right one asking at most p_2 questions.

iii) If the answer to question Q2 contains $p_1 - 1$ blacks, the secret contains b at exactly one of the pegs $1, \dots, p_1$. There are exactly p_1 secrets consistent with the answers. We find the right one asking at most p_1 questions.

All in all we obtain for the worst case

$$\begin{aligned} f(p,2) &\le \max\{p_1, p_2\} + 2 \\ &= \lceil (p-1)/2 \rceil + 2 \\ &= \lfloor p/2 \rfloor + 2. \qquad \square \end{aligned}$$

5. The general case

Combining former results we claim the following.

Theorem 5 *For $c \ge 2$ and $p \ge 2$ it holds*

a) $f(p,c) \ge \lfloor \frac{c-2}{p} \rfloor + \frac{p}{\log_2 p}$,

b) $f(p,c) \le \lfloor \frac{c-1}{p} \rfloor + 2p\lceil \log_2 p \rceil + 1$.

PROOF. To prove the lower bound we put $c_0 = 2$ in equation (6). We receive $f(p,c) \ge \lfloor (c-2)/p \rfloor + f_*(p,2)$. Observe that Theorem 3 gives also a lower bound for $f_*(p,2)$. We only need to argue that a property similar to equation (8) still holds. For f_* we have $W = \min\{k, k'\} + \min\{p-k, l'\} - B$, where l' is the number of colors b in the question. Hence W is still determined by B. This finishes the proof of the lower bound.

To prove the upper bound we show a strategy carrying it out. We start by asking $\lfloor (c-1)/p \rfloor$ questions each containing different p colors. We obtain at most p non-empty answers. If there are exactly p such answers, then all colors in the secret are inside adequate questions. If there are less than p non-empty answers, then we have to consider the remaining $r = c - p\lfloor (c-1)/p \rfloor$ colors, which we have not asked for. Note that $1 \le r \le p$. So we have at most p groups of colors, in which we have to search for colors in the secret. Each group contains p colors, except possibly the last special group with r colors. This means that we limited the number of possible secret colors to at most p^2. Let m be the number of all white and black marks received up to now. Obviously, $m \le p$. Note that the last special group contains at most $p - m$ secret colors.

After that we do a two-phase binary search procedure. In the first phase we search for colors which are in the secret. We want to limit the number of possible colors to at most p. In the second phase we find the right position(s) for each color.

The first phase is shown in Figure 1. The set C collects colors which can be in the secret. The outer loop iterates by groups of colors. For each group g the set A_g contains the colors from the group and k_g is the number of white and black marks obtained for the group in the adequate answer, except the

Fig. 1. The first phase of the binary search

```
C := ∅
for each group of colors g do
begin
  repeat k_g times
  begin
    X := A_g \ C
    while |X| > 1 do
    begin
      Split X into two disjoint subsets X_1, X_2
      with ⌈|X|/2⌉, ⌊|X|/2⌋ colors, respectively.
      Ask an arbitrary question which contains
      all colors from X_1, but no further colors.
      if non-empty answer then X := X_1
      else X := X_2
    end
    C := C ∪ X
  end
end
```

last special group. For this group we did not ask any question and we set $k_g = \min\{r, p-m\}$. After each iteration the set C increases by k_g colors. Adding one color to the set C costs $\lceil \log_2 p \rceil$ questions. At the end of the procedure the set C contains at most p colors and all colors from the secret are in C. The total cost of the first phase is at most $p\lceil \log_2 p \rceil$ questions.

We start the second phase by asking question $xxx\ldots x$ for each color $x \in C$ except the last one. This costs at most $p-1$ questions. The received number of black marks is the repetition factor of the color in the secret. We do not need to ask for the last color, because its repetition factor can be deduced by the previous answers.

If $c > p$ we know one color which is not in the secret, say z. Now for each color $x \in C$ with corresponding repetition factor we find the right position using binary search. For already known positions we use known colors and for the rest we use the colors x and z. This costs at most

$$\sum_{i=1}^{p} \lceil \log_2 i \rceil \leq p\lceil \log_2 p \rceil - p + 1 \qquad (9)$$

questions, where (9) is not hard to show. The total cost of the second phase is at most $p\lceil \log_2 p \rceil$ questions.

If $c \leq p$ we do not need the first phase of the binary search. We choose the set C containing all colors. After the first $p-1$ questions of the second phase we know how many times each color appears in the secret. If there is a color which is not in the secret we have our color z. If there is no such color we choose two colors, say a and b. We are able to find all positions of color a asking p questions: $ab\ldots bb$, $ba\ldots bb$, $\ldots$, $bb\ldots ab$, $bb\ldots ba$. For the remaining positions we do the second phase of the binary search putting a on all its right positions and using a as color z. This keeps us within the bound, because it costs at most $(p-1) + p + (p\lceil \log_2 p \rceil - p + 1) \leq 2p\lceil \log_2 p \rceil$ questions.

To finish the game we need to ask the final question, which will be answered with p black marks. □

6. Summary and suggestions for future research

In this paper we computed new values $f(p,c)$ for ten pairs (p,c). Furthermore we presented a general formula for $f(3,c)$ and a tight lower and upper bound for $f(4,c)$. For $f(p,2)$ we gave an upper bound and a lower bound. We also found the first bounds for the general case $f(p,c)$. Most of these results admit improvements.

We hope that by speeding-up our program we can find an exact formula for $f(4,c)$. However using the presented method to obtain a formula for $f(5,c)$ seems to be difficult, because of the exponential nature of the problem. It is an open question, whether it holds $f(k,2) = f(2,k)$ for $k \geq 2$. Finally we suggest to transform our methods to the static variant of Mastermind.

References

[1] L. Bento, L. Pereira, A.C. Rosa: Mastermind by Evolutionary Algorithms. Proc. of ACM Symp. Applied Computing, *ACM Press*, 307–311, 1999.

[2] S.T. Chen, S.S. Lin, L.T. Huang: Optimal Algorithms for $2 \times n$ Mastermind Games – a Graph Partition Approach. *Comput. J.* **47**(5), 602–611, 2004.

[3] S.T. Chen, S.S. Lin, L.T. Huang: A Two-Phase Optimization Algorithm for Mastermind. *Comput. J.* **50**(4), 435–443, 2007.

[4] W. Goddard: Static Mastermind. *J. Combin. Math. Combin. Comput.* **47**, 225–236, 2003.

[5] W. Goddard: Mastermind Revisited. *J. Combin. Math. Combin. Comput.* **51**, 215–220, 2004.

[6] T. Kalisker, D. Camens: Solving Mastermind Using Genetic Algorithms. Proc. of Genetic and Evolutionary Computation Conference (GECCO), *Lecture Notes in Comput. Sci.* **2724**, 1590–1591, 2003.

[7] D.E. Knuth: The Computer as Mastermind. *J. Recr. Math.* **9**, 1–6, 1976.

[8] K. Koyama, T.W. Lai: An Optimal Mastermind Strategy. *J. Recr. Math.* **25**(4), 251–256, 1993.

[9] B.D. McKay: Isomorph-Free Exhaustive Generation. *J. Algorithms* **26**, 306–324, 1998.

[10] J.J. Merelo-Guervós, P. Castillo, V.M. Rivas: Finding a Needle in a Haystack Using Hints and Evolutionary Computation: the Case of Evolutionary Mastermind. *Appl. S. Comput.* **6**, 170–179, 2006.

[11] J. Stuckan, G.Q. Zhang: Mastermind is NP-Complete. *INFOCOMP J. Comput. Sci.* **5**, 25–28, 2006.

[12] Source code of [9]. Available: "http://cs.anu.edu.au/people/bdm/nauty/".