Theoretical Computer Science 596 (2015) 55–66

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Bounding memory for Mastermind might not make it harder ☆

Gerold Jäger [a], Marcin Peczarski [b,*]

[a] Department of Mathematics and Mathematical Statistics, University of Umeå, SE-901-87 Umeå, Sweden
[b] Institute of Informatics, University of Warsaw, ul. Banacha 2, PL-02-097 Warszawa, Poland

ARTICLE INFO

Article history:
Received 10 November 2014
Received in revised form 12 April 2015
Accepted 18 June 2015
Available online 26 June 2015
Communicated by H.J. van den Herik

Keywords:
Game theory
Logic game
Mastermind
Space complexity

ABSTRACT

We investigate a version of the Mastermind game, where the codebreaker may only store a constant number of questions and answers, called Constant-Size Memory Mastermind, which was recently introduced by Doerr and Winzen. We concentrate on the most difficult case, where the codebreaker may store only one question and one answer, called Size-One Memory Mastermind. We consider two variants of the game: the original one, where the answer is coded with white and black pegs, and the simplified one, where only black pegs are used in the answer. We show that for two pegs and an arbitrary number of colors, the number of questions needed by the codebreaker for an optimal strategy in the worst case for these games is equal to the corresponding number of questions in the games without a memory restriction. We show that this also holds for the black-peg variant and three pegs. In other words, for these cases restricting the memory size does not make the game harder for the codebreaker. This is a continuation of a result of Doerr and Winzen, who showed that this holds asymptotically for a fixed number of colors and an arbitrary number of pegs. Furthermore, by computer search we determine additional pairs (p, c), where again the numbers of questions for an optimal strategy in the worst case for Size-One Memory Mastermind and original Mastermind are equal.

1. Introduction

Mastermind is a classical two-player board game invented by Meirowitz in 1970. The two players are usually called *codemaker* and *codebreaker*. The codemaker chooses a secret code consisting of 4 pegs and 6 possible colors for each peg. The codebreaker does not know this code and has to ask several questions about the code, until he (or she) has found the correct secret. The codemaker has to evaluate each question by giving an answer consisting of black and white pegs, where each black peg corresponds to a peg of the codebreaker's question which is correct in position and color, and each white peg corresponds to another peg which is correct only in color. Colors are usually coded by digits. For example, for the code 2345 and the question 2214 the answer would contain one black and one white peg. Naturally, the codebreaker's aim is to minimize the number of questions needed to find the secret. There are two ways of optimizing the codebreaker's strategy, namely minimizing the number of questions on *average* and minimizing the number of questions in the *worst case*, respectively. Both optimum strategies have been found for original Mastermind with 4 pegs and 6 colors using $5625/1296 \approx 4.34$ questions [15] and 5 questions [14], respectively.

☆ A preliminary version of this paper appeared in the proceedings of the 25th International Workshop on Combinatorial Algorithms (IWOCA), Duluth, USA, Oct. 15–17, 2014, see [12].

* Corresponding author.
E-mail addresses: gerold.jaeger@math.umu.se (G. Jäger), marpe@mimuw.edu.pl (M. Peczarski).

http://dx.doi.org/10.1016/j.tcs.2015.06.039
0304-3975/

Clearly, original Mastermind can easily be extended to *Generalized Mastermind* with p pegs and c colors. Stuckman and Zhang showed that the decision problem corresponding to Generalized Mastermind, i.e., the problem, whether a secret exists which satisfies a given set of questions and answers, is $\mathcal{NP}$-complete [16]. Let $f(p,c)$ be the number of questions needed by the codebreaker for an optimal strategy in the worst case for this game. Chen and Lin theoretically determined all values $f(2,c)$ to be equal to $\lfloor c/2 \rfloor + 2$ for $c \geq 2$ [2]. We determined a similar formula for $f(3,c)$ and proved a tight upper and lower bound for $f(4,c)$ and an upper and a lower bound for $f(p,2)$ [10]. Furthermore, we obtained additional values $f(p,c)$ by a computer search. Recently, Doerr et al. have shown that $f(c,c) = \mathcal{O}(c \log\log c)$ [4], improving the classical result $f(c,c) = \mathcal{O}(c \log c)$ of Chvátal [3]. Also many variants of Mastermind have been investigated in the literature. One important variant is black-peg Mastermind, where the codemaker only gives black-peg information and not white-peg information [9,11]. Let $b(p,c)$ be the number of questions needed by the codebreaker for an optimal strategy in the worst case for this variant of the game. We proved, among others, the exact formula $b(p,c) = c + p - 1$ for $p = 2, 3$ and $c \geq 2$ [11]. Other variants are static Mastermind [8] and the AB game [1,13].

In this work we investigate another version of the Mastermind game, which has recently been introduced by Doerr and Winzen [5,6]. They add to original Generalized Mastermind a so-called size-m memory restriction for the codebreaker. Then the codebreaker can store only up to m questions and the codemaker's corresponding answers and he can decide about the next question based only on this information. However, after receiving the answer, the codebreaker can decide which m of the last $m+1$ questions and corresponding answers he can store. For $m = 1$ this means that the codebreaker has a memory for storing only one pair of question and answer. Based only on this information he chooses the next question. After receiving the answer, he has two pairs of question and answer, and he decides which one to keep. The other one is discarded. Our codebreaker's strategy is even more restrictive. The codebreaker asks the question Q_i and remembers it. After receiving the answer A_i he remembers it and then decides about the next question Q_{i+1}. He asks and remembers Q_{i+1} and immediately forgets the question Q_i and the answer A_i. After receiving the answer A_{i+1} he remembers also A_{i+1}.

It should be noted that the memory, as defined in the work of Doerr and Winzen and in this paper, is in fact not constant. Memorizing a question requires $\Theta(p \log c)$ bits, and memorizing an answer requires $\Theta(\log p)$ bits. Hence, the memory increases linearly in p and logarithmically in c.

Let $f_m(p,c)$ and $b_m(p,c)$ be the number of questions needed by the codebreaker for an optimal strategy in the worst case for Size-m Memory Mastermind and its black-peg variant, respectively. Doerr and Winzen proved that for a fixed number of colors c the codebreaker has a size-one memory strategy winning the Mastermind game with p pegs using $\mathcal{O}(p/\log p)$ questions in the worst case, in other words, $f_1(p,c) = \mathcal{O}(p/\log p)$. A lower bound of $\Omega(p/\log p)$ was previously known, even without memory restriction, and it follows from an information theoretic argument [3,7]. Doerr and Winzen considered also the black-peg variant of this game and proved $b_1(p,c) = \mathcal{O}(p/\log p)$ for a fixed number of colors.

In this work we show an *exact* formula for the number of questions needed by the codebreaker for an optimal strategy in the worst case for Size-m Memory Mastermind with two pegs and for the black-peg variant with two or three pegs. In other words, we fix the number of pegs and not the number of colors. Our main results can be summarized in the following theorems.

Theorem 1. *The codebreaker has a size-one memory strategy winning the Mastermind game with two pegs and $c \geq 2$ colors using exactly $\lfloor c/2 \rfloor + 2$ questions in the worst case. Thus, it holds that $f_1(2,c) = f(2,c)$ for all c.*

Theorem 2. *The codebreaker has a size-one memory strategy winning the black-peg variant of the Mastermind game with two or three pegs and $c \geq 2$ colors using exactly $c + p - 1$ questions in the worst case. Thus, it holds that $b_1(p,c) = b(p,c)$ for $p = 2, 3$ and all c.*

This paper is organized as follows. In Section 2 we present the algorithm for Theorem 1, and we do a state analysis in Section 3, which finishes the proof of Theorem 1. Then we present the proof of Theorem 2 for $p = 2$ in Section 4 and for $p = 3$ in Section 5. The lower bounds of f_1 and b_1 are implied by the results of [10,11]. We describe additional results with more than two pegs in Section 6, where we use a modification of the program developed in [10]. We present some conclusions and a few suggestions for future research in Section 7. A preliminary version of the results presented in this paper appeared in [12].

Let xB denote the answer with x black pegs and no white pegs, and by xW the answer with x white pegs and no black pegs. Furthermore, let $\emptyset$ and 0B denote the zero (empty) answer. Observe that for the two pegs game an answer with one black peg and one white peg is impossible.

2. Algorithm for two pegs

A strategy for two-pegs Mastermind can be found in [10]. It starts with the questions $(0,1)$, $(2,3)$, $(4,5)$, i.e., we use questions of the pattern $(2i, 2i+1)$. At most two of them can receive a non-empty answer. The codebreaker needs to memorize these two questions and the associated answers. If the codebreaker is allowed to memorize only one question and answer, we use the following idea. We start asking questions as in [10], but after the first non-empty answer we change the pattern of asked questions so that it codes this question and its corresponding answer. Further questions are chosen such that a question codes all information about the current game state and needed to guess the secret. The key idea is to

code and recode a history of questions and answers into the current question, leading to a sequence of further questions. The algorithm is divided in five phases. Let $k = \lfloor c/2 \rfloor$. Note that in the following all values of i and j determine the game state. These values can always be computed from the memorized question and the number of colors c. Hence, we are always able to determine in which state we are.

Phase 1. We ask successively at most $k-2$ questions of the form $(2i, 2i+1)$ for $0 \le i \le k-3$, in increasing order of i. If the answers to all these questions are empty, then we go to Phase 3.

Otherwise, let i be the integer such that the first question with a non-empty answer is $(2i, 2i+1)$. We consider several cases:

- $(2i, 2i+1)$ receives the answer 2B: The game ends.
- $(2i, 2i+1)$ receives the answer 2W: We ask the question $(2i+1, 2i)$, which is answered by 2B, and the game ends.
- $(2i, 2i+1)$ receives the answer 1W: We go to Phase 2a.
- $(2i, 2i+1)$ receives the answer 1B: We go to Phase 2b.

Phase 2a. Up to now $i+1$ questions have been asked, and the question $(2i, 2i+1)$ is answered by 1W in Phase 1. Now we ask at most $k-i-1$ questions of the pattern

$$(2i+2, l), (2i+3, l+1), \ldots, (l-1, 2k-1),$$

where l will be chosen in the following. Note that the second color in the first question is the successor of the first color in the last question. Since the following relation must hold $(l-1) - (2i+2) = (2k-1) - l$, we have $l = k+i+1$. Thus, we ask the questions

$$(2i+2, k+i+1), (2i+3, k+i+2), \ldots, (k+i, 2k-1).$$

Observe that the answers 2B and 2W are impossible in this phase. Hence, we consider three cases:

- If the first j questions receive the empty answer and the question $(2i+j+2, k+i+j+1)$ is answered by 1W, where $j = 0, 1, 2, \ldots, k-i-2$, we go to Phase 4a.
- If the first j questions receive the empty answer and the question $(2i+j+2, k+i+j+1)$ is answered by 1B, where $j = 0, 1, 2, \ldots, k-i-2$, we go to Phase 4b.
- If all $k-i-1$ questions receive the empty answer, we go to Phase 5a.

Phase 2b. Up to now $i+1$ questions have been asked, and the question $(2i, 2i+1)$ is answered by 1B in Phase 1. Now we ask at most $k-i-1$ questions of the pattern

$$(k+i+1, 2i+2), (k+i+2, 2i+3), \ldots, (2k-1, k+i).$$

Note that the above questions differ from the questions of Phase 2a only by reversing the order of the colors. As previously, the answers 2B and 2W are impossible in this phase, and we consider three cases:

- If the first j questions receive the empty answer and the question $(k+i+j+1, 2i+j+2)$ is answered by 1W, where $j = 0, 1, 2, \ldots, k-i-2$, we go to Phase 4c.
- If the first j questions receive the empty answer and the question $(k+i+j+1, 2i+j+2)$ is answered by 1B, where $j = 0, 1, 2, \ldots, k-i-2$, we go to Phase 4d.
- If all $k-i-1$ questions receive the empty answer, we go to Phase 5b.

Let (a, b) denote the last question asked in Phase 2. The values of a and b encode the current game state. If $a < b$, then Phase 2a has been played and we have that $a = 2i+j+2$ and $b = k+i+j+1$. Hence, the values of i and j can be decoded as

$$i = a - b + k - 1,$$
$$j = -a + 2b - 2k.$$

If $a > b$, then Phase 2b has been played and we have that $a = k+i+j+1$ and $b = 2i+j+2$. Hence, the values of i and j can be decoded as

$$i = -a + b + k - 1,$$
$$j = 2a - b - 2k.$$

Phase 3. All $k-2$ first questions receive the empty answer. If c is even, then there are four possible colors: $c-4$, $c-3$, $c-2$, $c-1$. They can be denoted also as $2k-4$, $2k-3$, $2k-2$, $2k-1$. If c is odd, then there are five possible colors: $c-5$,

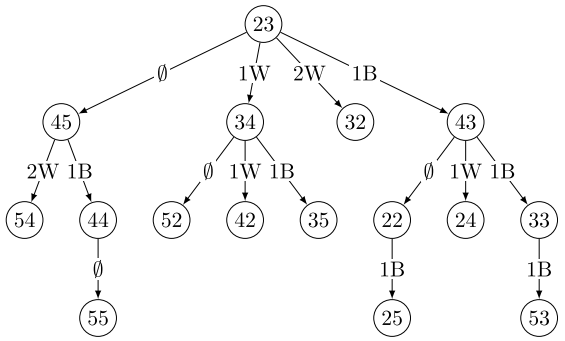

Fig. 1. Phase 3 for $c = 6$ colors.

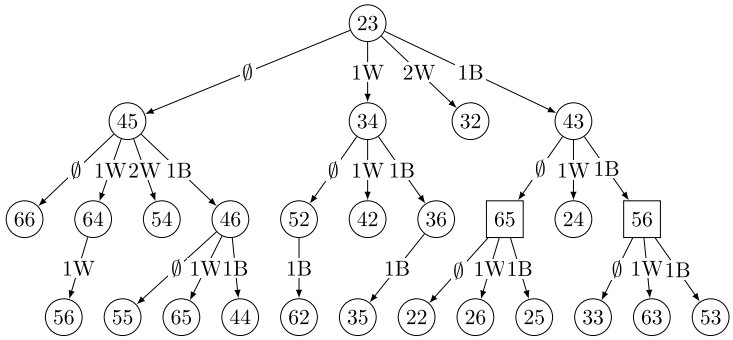

Fig. 2. Phase 3 for $c = 7$ colors.

$c-4$, $c-3$, $c-2$, $c-1$. They can be denoted also as $2k-4$, $2k-3$, $2k-2$, $2k-1$, $2k$. In both cases we can find the secret with four questions using only colors from the set of possible colors. Fig. 1 shows the strategy for Phase 3 when the game was started with $c = 6$ colors, and 4 colors (namely the colors 2, 3, 4, 5) remain possible for that phase. Fig. 2 shows the strategy for Phase 3, when the game was started with $c = 7$ colors, and 5 colors (namely the colors 2, 3, 4, 5, 6) remain possible for that phase. The nodes represent questions and the edges represent answers. Questions which can be answered by 2B are drawn in a circle, whereas questions which cannot be answered by 2B are drawn in a square. Note that questions in leafs are always answered by 2B. If c is even, then the strategy shown in Fig. 1 can be transformed into a general strategy by the color mapping $2 \mapsto 2k-4$, $3 \mapsto 2k-3$, $4 \mapsto 2k-2$, $5 \mapsto 2k-1$. If c is odd, then the strategy illustrated in Fig. 2 can be transformed into a general strategy by the above color mapping with the additional mapping $6 \mapsto 2k$.

Phase 4a. The question $(2i, 2i+1)$ is answered by 1W in Phase 1. The question $(2i+j+2, k+i+j+1)$ is answered by 1W in Phase 2a. Hence, there are two possible secrets, namely

$$(2i+1, 2i+j+2), (k+i+j+1, 2i).$$

We ask the question $(2i+1, 2i+j+2)$, and we consider two cases:

- If the answer is 2B, the game ends.
- If the answer is empty, we ask the question $(k+i+j+1, 2i)$, which is answered by 2B.

Note that all four colors used in this phase (and all Phases 4) are distinct, because $2i \neq k+i+j+1$, $2i+1 \neq k+i+j+1$ and $2i+j+2 \neq k+i+j+1$.

Phase 4b. The question $(2i, 2i+1)$ is answered by 1W in Phase 1. The question $(2i+j+2, k+i+j+1)$ is answered by 1B in Phase 2a. Hence, there are two possible secrets, namely

$$(2i+1, k+i+j+1), (2i+j+2, 2i).$$

We ask the same question as in Phase 4a, i.e., $(2i+1, 2i+j+2)$, and we consider two further cases:

- If the answer is 1W, we ask the question $(2i+j+2, 2i)$.
- If the answer is 1B, we ask the question $(2i+1, k+i+j+1)$.

In both cases the latter question is answered by 2B.

Phase 4c. The question $(2i, 2i+1)$ is answered by 1B in Phase 1. The question $(k+i+j+1, 2i+j+2)$ is answered by 1W in Phase 2b. Hence, there are two possible secrets, namely

$$(2i, k+i+j+1), (2i+j+2, 2i+1).$$

We ask the question $(2i+j+2, 2i+1)$, and we consider two cases:

- If the answer is 2B, the game ends.
- If the answer is empty, we ask the question $(2i, k+i+j+1)$, which is answered by 2B.

Phase 4d. The question $(2i, 2i+1)$ is answered by 1B in Phase 1. The question $(k+i+j+1, 2i+j+2)$ is answered by 1B in Phase 2b. Hence, there are two possible secrets, namely

$$(2i, 2i+j+2), (k+i+j+1, 2i+1).$$

We ask the same question as in Phase 4c, i.e., $(2i+j+2, 2i+1)$, and we consider two further cases:

- If the answer is 1W, we ask the question $(2i, 2i+j+2)$.
- If the answer is 1B, we ask the question $(k+i+j+1, 2i+1)$.

In both cases the latter question is answered by 2B.

Phase 5a. The question $(2i, 2i+1)$ is answered by 1W in Phase 1. All other questions receive the empty answer. It is impossible to reach this phase if c is even, because all colors except $2i$ and $2i+1$ are impossible.

If c is odd, then there are two possible secrets, namely

$$(c-1, 2i), (2i+1, c-1).$$

We ask the question $(c-1, 2i)$, and we consider two cases:

- If the answer is 2B, the game ends.
- If the answer is 1W, we ask the question $(2i+1, c-1)$, which is answered by 2B.

Phase 5b. The question $(2i, 2i+1)$ is answered by 1B in Phase 1. All other questions receive the empty answer. If c is even, then there are two possible secrets, namely

$$(2i, 2i), (2i+1, 2i+1).$$

We ask the question $(2i, 2i)$, and we consider two cases:

- If the answer is 2B, the game ends.
- If the answer is empty, we ask the question $(2i+1, 2i+1)$, which is answered by 2B.

If c is odd, then there are four possible secrets, namely

$$(2i, 2i), (2i+1, 2i+1), (2i, c-1), (c-1, 2i+1).$$

We ask the question $(2i, c-1)$, and we consider four cases:

- If the answer is 1B, we ask the question $(2i, 2i)$, which is answered by 2B.
- If the answer is empty, we ask the question $(2i+1, 2i+1)$, which is answered by 2B.
- If the answer is 2B, the secret is $(2i, c-1)$, and the game ends.
- If the answer is 1W, we ask the question $(c-1, 2i+1)$, which is answered by 2B.

Small number of colors. If the number of colors is 4 or 5, then Phase 1 reduces to zero questions and thus there is also no Phase 2, 4 and 5. The whole algorithm reduces to Phase 3 only, using at most 4 questions. If the number of colors is 2 or 3, then we use the part of Phase 3, which does not ask the first question, and assume that it receives the empty answer. In those cases we find a secret in at most 3 questions. Thus, we have $f_1(2, c) \le k+2$ for $2 \le c \le 5$. It is easy to see that $f_1(2, 1) = f(2, 1) = 1$.

At the end of this section, we count the number of questions needed to find a secret for $c \ge 6$. Phase 1 uses $i+1 \le k-2$ questions. Phase 2 uses at most $k-i-1$ questions. Phase 3 uses at most four questions. Phases 4 and 5 use at most two questions. Fig. 3 shows all possible phase transitions. This leads to the following three possible scenarios:

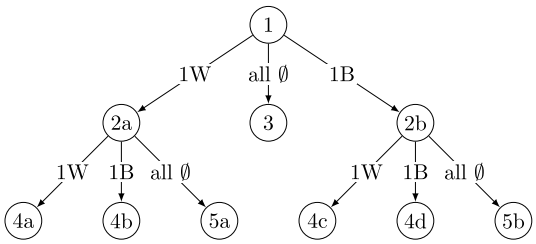

Fig. 3. Phase transition diagram.

- Phase 1, Phase 2, Phase 4;
- Phase 1, Phase 2, Phase 5;
- Phase 1, Phase 3.

It is easy to see that we find a secret in at most $k+2$ questions, i.e., $f_1(2,c) \leq k+2$.

It holds that $f_1(2,c) \geq f(2,c)$ and it holds by [10] that $f(2,c) = k+2$ for $c \geq 2$. Finally, we have that $f_1(2,c) = f(2,c)$ for all c.

3. State analysis for two pegs

We need to check if the above algorithm states a strategy for the codebreaker with size-one memory. It is sufficient to prove that the asked question depends only on the previous question and the received answer. This will be implied by the property that either a question is used only in one game state or if a question is used in two game states, then the sets of answers in these states are disjoint.

The sets of questions asked in Phases 1, 2a, 2b are disjoint, because the colors used in a question in Phases 2a and 2b differ by at least two, as

$$(k+i+1) - (2i+2) = k-i-1 \geq k-(k-3)-1 = 2.$$

Moreover, in Phase 2a the first color in the question has a smaller number than the second one, and in Phase 2b the first color in the question has a larger number than the second one.

Phase 3 uses distinct colors compared to Phase 1. Hence, it uses also distinct questions. Note that the questions $(2k-4, 2k-2)$, $(2k-3, 2k-1)$, $(2k-2, 2k-4)$, $(2k-1, 2k-3)$ are used only if they are expected to be answered by 2B, as they are also used in Phase 2a or 2b, where a question cannot be answered by 2B.

As the second question in Phase 4 is answered only by 2B, it can be used in all other phases. The first question of Phase 4 is distinct from all questions of Phase 1, because:

- the first color has a smaller number than the second one and the first one is odd, e.g., the question $(2i+1, 2i+j+2)$, or
- the first color has a larger number than the second one and the second one is odd, e.g., the question $(2i+j+2, 2i+1)$.

Phase 4 uses distinct questions compared to Phase 3, because one color in each question of Phase 4 has a number smaller than $2k-4$ (this is the color which was used in Phase 1).

In the following we will prove that the first question used in Phase 4 is not used in Phase 2. The first question used in Phase 4a or 4b cannot be used in Phase 2b, because it has an opposite color number order. Now assume that the first question used in Phase 4a or 4b is also used in Phase 2a. Let $(2i'+j'+2, k+i'+j'+1)$ be a question asked in Phase 2a. Let $(2i+1, 2i+j+2)$ be the first question asked in Phase 4a or 4b. This implies that

$$2i'+j'+2 = 2i+1, \tag{1}$$

$$k+i'+j'+1 = 2i+j+2. \tag{2}$$

Hence, we have $2(k+i'+j'+1) - (2i'+j'+2) = 2(2i+j+2) - (2i+1)$ and $2k+j' = 2i+2j+3$. However, $i+j \leq k-2$ and thus $j' \leq -1$, which is a contradiction.

A similar result can be proved for Phases 4c, 4d and Phases 2a, 2b as follows. The first question used in Phase 4c or 4d cannot be used in Phase 2a, because it has an opposite color number order. Now assume that the first question used in Phase 4c or 4d is also used in Phase 2b. Let $(k+i'+j'+1, 2i'+j'+2)$ be a question asked in Phase 2b. Let $(2i+j+2, 2i+1)$ be the first question asked in Phase 4c or 4d. This implies exactly the same equations (1) and (2) as previously, and we have a contradiction again.

The questions used in Phase 5a or 5b are not used in Phase 3, because $2i < 2k-4$ and $2i+1 < 2k-4$ holds. They are also not used in any other phase, because questions can contain the color $2k$ (i.e., color $c-1$ for odd c) or twice the same color only in Phases 3 and 5.

This finishes the proof of correctness and of Theorem 1.

4. Algorithm for black-peg variant and two pegs

Let us consider a Mastermind strategy, where we choose only questions for which the answer pB (p black pegs) is possible. In other words, we choose questions only from the set of secrets which are still possible based on all answers given yet by the codemaker. We call such a strategy a P-strategy.

In this and the following section we will need the following result:

Proposition 1. *Every P-strategy is a size-one memory strategy.*

Proof. Every strategy-tree for Mastermind has to contain as nodes all possible questions, because for every possible secret there must be a question equal to this secret and answered by p black pegs. Every question appears at most once in the P-strategy-tree, because distinct paths in the strategy-tree lead to disjoint sets of possible secrets. Hence, every P-strategy-tree contains exactly c^p nodes, where each node is labeled with another question. Therefore, each question appears exactly once. Moreover, a P-strategy-tree has exactly $c^p - 1$ edges. In general, to store the game state, the codebreaker has to remember the whole path from the root to the present node, but in the P-strategy the label of the node (a question) uniquely determines the game state. Hence, the codebreaker has to memorize only one question to store a position in the strategy-tree, and the P-strategy-tree immediately defines a strategy for Size-One Memory Mastermind. □

A strategy for the black-peg variant of two-pegs Mastermind can be found in [11]. We adapt this strategy to a P-strategy. Note that in this game we have only three possible answers, namely 0B, 1B and 2B.

Consider the following strategy for $c \geq 2$. We ask successively at most $c-1$ questions of the form (i,i) for $0 \leq i \leq c-2$, in increasing order of i. If the answers of all these questions are empty, then the only possible secret is $(c-1, c-1)$. We ask the question $(c-1, c-1)$, which is answered by 2B. Thus, we are done in c questions. Otherwise, let i be the integer such that the first question with a non-empty answer is (i,i). We consider three cases:

1. (i,i) receives the answer 2B: We are done in $i+1 \leq c-1$ questions.
2. (i,i) receives the answer 1B: We ask the question $(i, i+1)$.
3. (i,i) receives the answer 0B: By construction, this case is not possible.

Regarding the question $(i, i+1)$ (Case 2) we have three next cases:

4. $(i,i+1)$ receives the answer 2B: We are done in $i+2 \leq c$ questions.
5. $(i,i+1)$ receives the answer 1B: Then $i < c-2$ and we have $c-i-2$ possible secrets: $(i,i+2), (i,i+3), \ldots, (i,c-1)$, which we can test in $c-i-2$ further questions. Then totally we have at most $i+2+c-i-2 = c$ questions.
6. $(i,i+1)$ receives the answer 0B: Then we ask the question $(i+1, i)$.

Regarding the question $(i+1, i)$ (Case 6) we have three further cases:

7. $(i+1,i)$ receives the answer 2B: We are done in $i+3 \leq c+1$ questions.
8. $(i+1,i)$ receives the answer 1B: Then $i < c-2$ and we have $c-i-2$ possible secrets: $(i+2,i), (i+3,i), \ldots, (c-1,i)$, which we can test in $c-i-2$ further questions. Totally, we have at most $i+3+c-i-2 = c+1$ questions.
9. $(i+1,i)$ receives the answer 0B: This case is not possible.

The algorithm is a P-strategy and by Proposition 1 also a size-one memory strategy. We conclude that $b_1(2,c) \leq c+1$. It holds that $b_1(2,c) \geq b(2,c)$ and it holds by [11] that $b(2,c) = c+1$ for $c \geq 2$. It is easy to see that $b_1(2,1) = b(2,1) = 1$. This finishes the proof of Theorem 2 for $p = 2$.

For a better understanding of the strategy, below we present a function for the codebreaker. The function takes a question and an answer (0B or 1B) and returns the next question. We omit the answer 2B, because after that the game ends.

$$
\begin{aligned}
(i,i), 0\text{B} &\mapsto (i+1, i+1) && \text{for } i = 0,1,\ldots,c-2,\\
(i,i), 1\text{B} &\mapsto (i, i+1) && \text{for } i = 0,1,\ldots,c-2,\\
(i,i+1), 0\text{B} &\mapsto (i+1, i) && \text{for } i = 0,1,\ldots,c-2,\\
(i,i+j+1), 1\text{B} &\mapsto (i, i+j+2) && \text{for } i = 0,1,\ldots,c-3,\ j = 0,1,\ldots,c-i-3,\\
(i+j+1,i), 1\text{B} &\mapsto (i+j+2, i) && \text{for } i = 0,1,\ldots,c-3,\ j = 0,1,\ldots,c-i-3.
\end{aligned}
$$

Note that the questions $(i, i+j+1)$ for $j > 0$ and $(i+j+1, i)$ for $j \geq 0$ cannot be answered by 0B.

5. Algorithm for black-peg variant and three pegs

In this section we prove Theorem 2 for $p = 3$ by showing a P-strategy. The algorithm is an extension of the case of two pegs and uses this case as a sub-procedure. For this algorithm we need the following auxiliary game. Let $p = 2$ and $c \geq 2$. Assume that the codebreaker knows that for the first peg only $c - 1$ colors are possible and for the second peg c colors. Consider the black-peg variant of this game. This is the same game as in Section 4, but in the first peg the color $c - 1$ is not possible any more.

Claim 1. *For the auxiliary game a P-strategy exists which needs only c (and not $c + 1$) questions.*

Proof. Note that only after the question $(i + 1, i)$ of Case 6 of Section 4 the situation occurs that we ask a question which has the first peg of color $c - 1$ or the situation that we need $c + 1$ (and not c) questions. We slightly change the strategy from Section 4 by asking in Case 8 only the sequence of questions $(i + 2, i)$, $(i + 3, i)$, …, $(c - 2, i)$ instead of $(i + 2, i)$, $(i + 3, i)$, …, $(c - 1, i)$.

Now consider two cases for the question (i, i) before Case 1:

- $i = c - 2$.
  Then the question $(i, i + 1)$ of Case 2 must be answered by 2B, i.e., the question $(i + 1, i)$ of Case 6 is not asked, or in other words, the Cases 7 and 8 are not reached in this case. Hence, in this case we need only c questions and we still have a P-strategy.
- $i < c - 2$.
  Then $i + 3 \leq c$, and in Case 7 we need only c questions. In Case 8 we do not have to ask the question $(c - 1, i)$ any more and so we save one question. Hence, again we need only c questions. As the question $(c - 1, i)$ is not asked, we still have a P-strategy.

This proves the claim. □

Now let us return to the black-peg variant of Mastermind for three pegs. In this game we have only four possible answers, namely 0B, 1B, 2B and 3B. In the following we present a P-strategy for $c \geq 2$.

We ask successively at most $c - 1$ questions of the form (i, i, i) for $0 \leq i \leq c - 2$, in increasing order of i. If the answers of all these questions are empty, then the only possible secret is $(c - 1, c - 1, c - 1)$. We ask the question $(c - 1, c - 1, c - 1)$, which is answered by 3B. Thus, we are done in c questions.

Otherwise, let i be the integer such that the first question with a non-empty answer is (i, i, i). Thus, in the following we assume that i is a fixed number. In this case we continue, following Fig. 4, and then Fig. 5 or Fig. 6. In these figures, ellipses contain questions, arrow labels represent answers, and rectangles describe sets of possible secrets. Furthermore, w, x, x', y, y', z, z' denote some unknown colors in the code so that in all figures it holds: $i + 1 \leq z, z' \leq c - 1$; $i + 2 \leq y, y' \leq c - 1$; $i + 3 \leq x, x' \leq c - 1$; $i + 4 \leq w \leq c - 1$. Special cases are listed below and are sorted in lexicographical order. The cases are numbered such that the number codes the answers given by the codemaker. The leftmost digit denotes the number of black pegs in the first non-empty answer. The next digits denote the number of black pegs in the subsequent answers.

Case 10: This case is shown in Fig. 5.

Case 100: Up to now we have asked $i + 3$ questions. Then the remaining game is equivalent to the black-peg variant with $p = 2$ pegs and $c' = c - i - 2$ colors. This game needs $c - i - 1$ further questions. Hence, totally we need at most $c + 2$ questions.

Case 1010: Up to now we have asked $i + 4$ questions. We have the possible secrets $(i + 1, y, i)$ with $i + 2 \leq y \leq c - 1$. Then we ask at most $c - i - 2$ questions: $(i + 1, i + 2, i)$, $(i + 1, i + 3, i)$, …, $(i + 1, c - 1, i)$. Totally, we need at most $c + 2$ questions.

Cases 1011 and 1012: Up to now we have asked $i + 4$ questions. In the second peg the color has to be i. We have the possible secrets (y, i, x), where $i + 2 \leq y \leq c - 1$ and $i + 3 \leq x \leq c - 1$. Reordering colors $c - 1 \mapsto 0$, $c - 2 \mapsto 1$, …, $i + 3 \mapsto c - i - 4$, $i + 2 \mapsto c - i - 3$ and omitting the second peg, we have precisely the introduced auxiliary game with $c' = c - i - 2$, where we have already asked one question $(0, 0)$ which is answered by 0B or 1B. As stated in the claim, we have to ask at most $c - i - 2$ questions for this auxiliary game. Totally, we need at most $(c - i - 2) + i + 4 - 1 = c + 1$ questions.

Case 1013: The game is finished. We have asked $i + 4$ questions. If we are in this case, then there exists the color $i + 2$ and $i + 2 \leq c - 1$. Hence, we have $i + 4 \leq c + 1$.

Case 1021: Up to now we have asked $i + 4$ questions. Furthermore, we ask at most $c - i - 3$ questions: $(i + 1, i, i + 3)$, $(i + 1, i, i + 4)$, …, $(i + 1, i, c - 1)$. Totally, we need at most $c + 1$ questions.

Case 1022: Up to now we have asked $i + 4$ questions. Furthermore, we ask at most $c - i - 3$ questions: $(i + 3, i, i + 2)$, $(i + 4, i, i + 2)$, …, $(c - 1, i, i + 2)$. Totally, we need at most $c + 1$ questions.

Case 1023: The game is finished. We have asked $i + 4$ questions. If we are in this case, then there exists the color $i + 2$ and $i + 2 \leq c - 1$. Hence, we have $i + 4 \leq c + 1$.

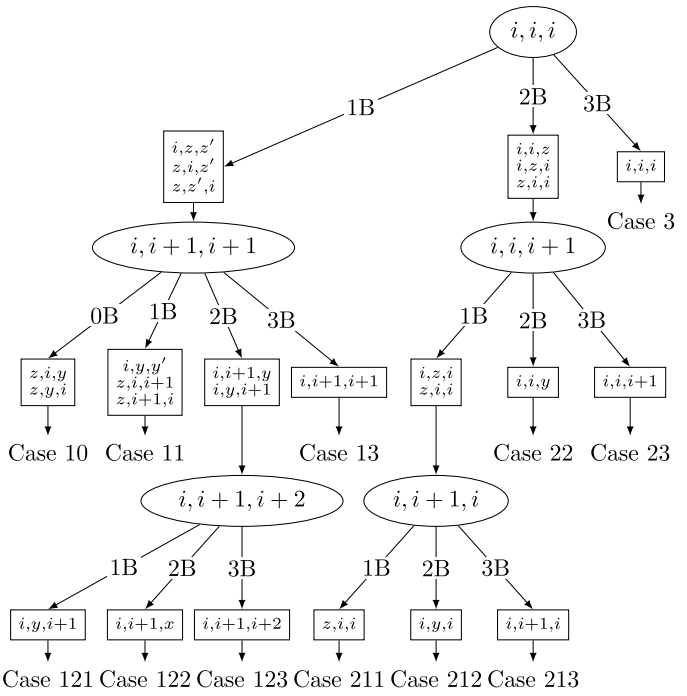

Fig. 4. Main strategy for the black-peg variant of Mastermind with size-one memory for three pegs.

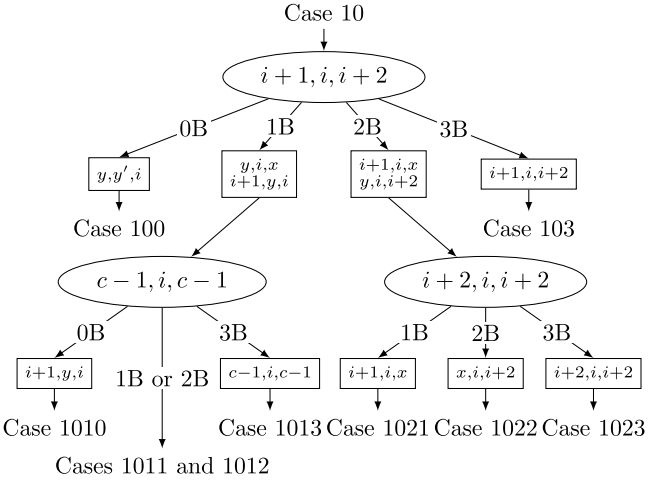

Fig. 5. Case 10 of the main strategy for the black-peg variant of Mastermind with size-one memory for three pegs.

Case 103: The game is finished. We have asked $i+3$ questions. If we are in this case, then there exists the color $i+2$ and $i+2 \le c-1$. Hence, we have $i+3 \le c$.

Case 11: This case is shown in Fig. 6.

Case 1100: Up to now we have asked $i+4$ questions. Furthermore, we ask at most $c-i-2$ questions: $(i+2, i+1, i)$, $(i+3, i+1, i)$, …, $(c-1, i+1, i)$. Totally, we need at most $c+2$ questions.

Case 1101: Up to now we have asked $i+4$ questions. Then the remaining game is equivalent to the black-peg variant with $p=2$ pegs and $c'=c-i-3$ colors. This game needs $c-i-2$ further questions. Hence, totally we need at most $c+2$ questions.

Case 11021: Up to now we have asked $i+5$ questions. Furthermore, we ask at most $c-i-3$ questions: $(i, i+3, i+2)$, $(i, i+4, i+2)$, …, $(i, c-1, i+2)$. Totally, we need at most $c+2$ questions.

Case 11022: Up to now we have asked $i+5$ questions. Furthermore, we ask at most $c-i-4$ questions: $(i, i+2, i+4)$, $(i, i+2, i+5)$, …, $(i, i+2, c-1)$. Totally, we need at most $c+1$ questions.

Case 11023: The game is finished. We have asked $i+5$ questions. If we are in this case, then there exists the color $i+3$ and $i+3 \le c-1$. Hence, we have $i+5 \le c+1$.

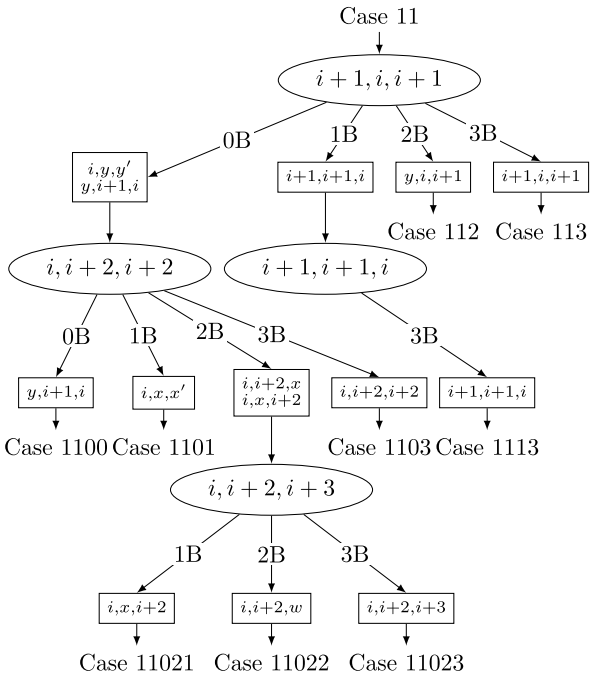

Fig. 6. Case 11 of the main strategy for the black-peg variant of Mastermind with size-one memory for three pegs.

Case 1103: The game is finished. We have asked $i+4$ questions. If we are in this case, then there exists the color $i+2$ and $i+2 \le c-1$. Hence, we have $i+4 \le c+1$.

Case 1113: The game is finished. We have asked $i+4$ questions. If we are in this case, then there exists the color $i+1$ and $i+1 \le c-1$. Hence, we have $i+4 \le c+2$.

Case 112: Up to now we have asked $i+3$ questions. Furthermore, we ask at most $c-i-2$ questions: $(i+2,i,i+1)$, $(i+3,i,i+1)$, …, $(c-1,i,i+1)$. Totally, we need at most $c+1$ questions.

Case 113: The game is finished. We have asked $i+3$ questions. If we are in this case, then there exists the color $i+1$ and $i+1 \le c-1$. Hence, we have $i+3 \le c+1$.

Case 121: Up to now we have asked $i+3$ questions. Furthermore, we ask at most $c-i-2$ questions: $(i,i+2,i+1)$, $(i,i+3,i+1)$, …, $(i,c-1,i+1)$. Totally, we need at most $c+1$ questions.

Case 122: Up to now we have asked $i+3$ questions. Furthermore, we ask at most $c-i-3$ questions: $(i,i+1,i+3)$, $(i,i+1,i+4)$, …, $(i,i+1,c-1)$. Totally, we need at most c questions.

Case 123: The game is finished. We have asked $i+3$ questions. If we are in this case, then there exists the color $i+2$ and $i+2 \le c-1$. Hence, we have $i+3 \le c$.

Case 13: The game is finished. We have asked $i+2$ questions. If we are in this case, then there exists the color $i+1$ and $i+1 \le c-1$. Hence, $i+2 \le c$.

Case 211: Up to now we have asked $i+3$ questions. Furthermore, we ask at most $c-i-1$ questions: $(i+1,i,i)$, $(i+2,i,i)$, …, $(c-1,i,i)$. Totally, we need at most $c+2$ questions.

Case 212: Up to now we have asked $i+3$ questions. Furthermore, we ask at most $c-i-2$ questions: $(i,i+2,i)$, $(i,i+3,i)$, …, $(i,c-1,i)$. Totally, we need at most $c+1$ questions.

Case 213: The game is finished. We have asked $i+3$ questions. If we are in this case, then there exists the color $i+1$ and $i+1 \le c-1$. Hence, we have $i+3 \le c+1$.

Case 22: Up to now we have asked $i+2$ questions. Furthermore, we ask at most $c-i-2$ questions: $(i,i,i+2)$, $(i,i,i+3)$, …, $(i,i,c-1)$. Totally, we need at most c questions.

Case 23: The game is finished. We have asked $i+2$ questions. If we are in this case, then there exists the color $i+1$ and $i+1 \le c-1$. Hence, we have $i+2 \le c$.

Case 3: The game is finished. We have asked $i+1$ questions. If we are in this case, then there exists the color i and $i \le c-1$. Hence, we have $i+1 \le c$.

Case 0 does not exist, because we consider only non-empty answers to the question (i,i,i). Clearly, also the cases 1020, 11020, 1110, 1111, 1112, 120, 20 and 210 are not possible.

The algorithm is a P-strategy and by Proposition 1 also a size-one memory strategy. We conclude that $b_1(3,c) \le c+2$. It holds that $b_1(3,c) \ge b(3,c)$ and by [11] it holds that $b(3,c) = c+2$ for $c \ge 2$. It is easy to see that $b_1(3,1) = b(3,1)$. This finishes the proof of Theorem 2 for $p = 3$.

6. More pegs

Using computer search, we can show that there are additional pairs (p, c), where the numbers of questions needed by the codebreaker for an optimal strategy in the worst case for Size-One Memory Mastermind and original Mastermind are equal, i.e., it holds that $f(p, c) = f_1(p, c)$ or $b(p, c) = b_1(p, c)$. For obtaining these results we have modified the program from [10] for computing the number of questions $f(p, c)$ needed by the codebreaker in an optimal strategy in the worst case for original Mastermind. As the program was not able to compute $f_1(p, c)$ or $b_1(p, c)$ directly, we have restricted the computations to find only P-strategies.

The program is parametrized with the variant of the game, the number of pegs p, the number of colors c and the worst case number of questions q. It outputs *true*, if there exists a worst case strategy for the codebreaker with at most q questions, and it outputs *false* otherwise. Let M denote the set of possible secrets. The program uses a recursive procedure which takes as input parameters the current set M and the number of remaining questions q. As the procedure searches for a P-strategy, it uses as questions all secrets from the set M. All answers are considered for every question. For each pair of question/answer the procedure computes the subset $M' \subseteq M$ of secrets consistent with this pair. The procedure is called recursively for every non-empty subset M' and $q' = q - 1$ questions. If there exists a question such that the result of a recursive call is *true* for every answer, the procedure returns *true*. Otherwise it returns *false*. The recursion starts with the full set of possible secrets. The recursion stops in two cases: the procedure returns *true*, if the answer is p blacks; the procedure returns *false*, if q is zero.

Let $f_0(p, c)$ and $b_0(p, c)$ be the worst case number of questions in a P-strategy for the original and black-peg variants, respectively. Clearly, it holds that $f_0(p, c) \geq f_1(p, c) \geq f(p, c)$ and $b_0(p, c) \geq b_1(p, c) \geq b(p, c)$. Hence, if we obtain that $f_0(p, c) = f(p, c)$ or $b_0(p, c) = f(p, c)$, we can conclude the value of $f_1(p, c)$ or $b_1(p, c)$.

The computed values are listed in the following. The values of $b_0(2, c)$ and $b_0(3, c)$ follow directly from Theorem 2 and thus they are omitted. The values of $f(p, c)$ and $b(p, c)$ are taken from [10,11].

$$
\begin{array}{cc}
f_0(2,2) = f(2,2) = 3 & f_0(4,2) = f(4,2) = 4 \\
f_0(2,3) = f(2,3) = 3 & f_0(4,3) = f(4,3) = 4 \\
f_0(2,4) = f(2,4) = 4 & f_0(4,4) = 5 > f(4,4) = 4 \\
f_0(2,5) = 5 > f(2,5) = 4 & f_0(4,5) = f(4,5) = 5 \\
f_0(2,6) = 6 > f(2,6) = 5 & f_0(4,6) = 6 > f(4,6) = 5 \\
f_0(3,2) = f(3,2) = 3 & f_0(5,2) = f(5,2) = 4 \\
f_0(3,3) = f(3,3) = 4 & f_0(5,3) = f(5,3) = 4 \\
f_0(3,4) = f(3,4) = 4 & f_0(5,4) = f(5,4) = 5 \\
f_0(3,5) = f(3,5) = 5 & f_0(6,2) = f(6,2) = 5 \\
f_0(3,6) = 6 > f(3,6) = 5 & f_0(6,3) = f(6,3) = 5
\end{array}
$$

Thus, we receive the additional pairs $(3, 2)$, $(3, 3)$, $(3, 4)$, $(3, 5)$, $(4, 2)$, $(4, 3)$, $(4, 5)$, $(5, 2)$, $(5, 3)$, $(5, 4)$, $(6, 2)$, $(6, 3)$ for which it holds that $f_1(p, c) = f(p, c)$.

$$
\begin{array}{cc}
b_0(4,2) = 5 = b(4,2) = 5 & b_0(5,2) = 6 > b(5,2) = 5 \\
b_0(4,3) = 9 > b(4,3) = 5 & b_0(5,3) = 9 > b(5,3) = 6 \\
b_0(4,4) = 10 > b(4,4) = 6 & b_0(6,2) = 7 > b(6,2) = 6
\end{array}
$$

Thus, we receive an additional pair $(4, 2)$ for which it holds that $b_1(p, c) = b(p, c)$.

7. Conclusions and future work

In this work we have obtained a result continuing the work of Doerr and Winzen [5,6], who showed that in the original Mastermind game for a fixed number of colors, bounding memory for the codebreaker does not increase the asymptotic number of questions in the worst case. We show a similar result for two and three pegs, i.e., here the number of pegs and not the number of colors is fixed. However, our result does not show only an asymptotic bound, but an exact one.

The computational results from Section 6 show that the approach used for the black-peg variant of Size-One Memory Mastermind with two or three pegs cannot be extended to a larger number of pegs, as for $p = 4, 5, 6$ there exists a c such that $b_0(p, c) > b(p, c)$, i.e., there is no appropriate P-strategy. There is also no hope that this approach can be used for original Mastermind, as for $p = 2, 3, 4$ we found a c such that $f_0(p, c) > f(p, c)$.

For future research we suggest to search for a pair (p, c), where the size-one memory version needs more questions than the version with unbounded memory. The first candidates are $(4, 4)$ for the original variant, as $f_0(4, 4) > f(4, 4)$, and $(4, 3)$ for the black-peg variant, as $b_0(4, 3) > b(4, 3)$. If we have found such a case, it would be interesting to consider strategies with size-two or larger memory.

References

[1] S.T. Chen, S.S. Lin, Optimal algorithms for $2 \times n$ AB games—a graph-partition approach, J. Inf. Sci. Eng. 20 (1) (2004) 105–126.

[2] S.T. Chen, S.S. Lin, Optimal algorithms for $2 \times n$ mastermind games—a graph-partition approach, Comput. J. 47 (5) (2004) 602–611.
[3] V. Chvátal, Mastermind, Combinatorica 3 (3–4) (1983) 325–329.
[4] B. Doerr, R. Spöhel, H. Thomas, C. Winzen, Playing mastermind with many colors, in: Proc. of 24th Annual ACM–SIAM Symposium on Discrete Algorithms, SODA 2013, SIAM, 2013, pp. 695–704.
[5] B. Doerr, C. Winzen, Playing mastermind with constant-size memory, in: Proc. of 29th International Symposium on Theoretical Aspects of Computer Science, STACS 2012, in: Leibniz International Proceedings in Informatics, LIPIcs, vol. 14, 2012, pp. 441–452.
[6] B. Doerr, C. Winzen, Playing mastermind with constant-size memory, Theory Comput. Syst. 55 (4) (2014) 658–684.
[7] P. Erdős, A. Rényi, On two problems of information theory, Magyar Tud. Akad. Mat. Kutató Int. Közl. 8 (1963) 229–243.
[8] W. Goddard, Static mastermind, J. Combin. Math. Combin. Comput. 47 (2003) 225–236.
[9] M.T. Goodrich, On the algorithmic complexity of the mastermind game with black-peg results, Inform. Process. Lett. 109 (13) (2009) 675–678.
[10] G. Jäger, M. Peczarski, The number of pessimistic guesses in generalized mastermind, Inform. Process. Lett. 109 (12) (2009) 635–641.
[11] G. Jäger, M. Peczarski, The number of pessimistic guesses in generalized black-peg mastermind, Inform. Process. Lett. 111 (19) (2011) 933–940.
[12] G. Jäger, M. Peczarski, Playing several variants of mastermind with constant-size memory is not harder than with unbounded memory, in: Proc. of 25th International Workshop on Combinatorial Algorithms, IWOCA 2014, in: Lecture Notes in Comput. Sci., vol. 8986, 2014, pp. 188–199.
[13] G. Jäger, M. Peczarski, The worst case number of questions in generalized AB game with and without white-peg answers, Discrete Appl. Math. 184 (2015) 20–31.
[14] D.E. Knuth, The computer as mastermind, J. Recreat. Math. 9 (1) (1976–1977) 1–6.
[15] K. Koyama, T.W. Lai, An optimal mastermind strategy, J. Recreat. Math. 25 (4) (1993) 251–256.
[16] J. Stuckman, G.Q. Zhang, Mastermind is NP-complete, INFOCOMP J. Comput. Sci. 5 (2006) 25–28.