Zeitschrift für Operations Research, Band 26, 1982, S. B257–B278.

Some Strategies for Mastermind

By *E. Neuwirth*, Wien[1])

Eingegangen am 2. Dezember 1981

Summary: The aim of this paper is to present some methods of construction for strategies for the game "Mastermind".

Some of these methods were already published before by other authors, but in this paper an approach unifying most of these methods is used. Numerical results for expectation and maximum number of questions needed to finish the game are presented for different versions of the game. By combining some of these methods a strategy slightly better than all strategies published by other authors is constructed for the most popular version of the game.

Zusammenfassung: Der vorliegende Artikel will mehrere Konstruktionsmethoden für Strategien für das Spiel „Mastermind" darstellen und miteinander vergleichen.

Einige dieser Methoden wurden bereits von anderen Autoren publiziert, der vorliegende Artikel beschreibt allerdings ein übergeordnetes Konzept, in das alle diese Methoden eingeordnet werden können. Numerische Ergebnisse für den Erwartungswert und die maximal notwendige Anzahl von Fragen bis zum Spielende werden für verschiedene Strategien und verschiedene Versionen dieses Spiels präsentiert. Durch Kombination mehrerer Methoden ist es sogar möglich, für die bekannteste Version dieses Spiels eine – wenn auch nur geringfügig – bessere Strategie als alle bisher publizierten anzugeben.

1. Introduction

Let us repeat the rules of the game in the standard version:
A secret code consisting of a fixed number of coloured pegs is to be guessed in the right order. For this purpose the player poses questions in form of combinations of pegs and gets the answer in form of two numbers: He is informed how many of his pegs are correct colours in the correct position and how many of his pegs are correct colours in wrong positions. His aim is to find out the secret code with a small number of questions. The transformation of this game into a notation fit for algorithmic treatment is trivial: Each colour is represented by a number. Hence, each colour-code is represented by a sequence of numbers.

---

[1]) Dr. *Erich Neuwirth*, Institut für Statistik der Universität Wien, Rathausstraße 19/4, A–1010 Wien.

0340-9422/82/08B257–B278$2.50

Now one may study the following problem: Assuming equal probabilities for all the possible codes find strategies with small expectation for the number of questions needed to break the code. This paper will study different types of strategies for different versions of the game. (Some of these types were already studied in *Irving* [1979].)

The second interesting problem is:
Find strategies which minimize the maximum number of questions needed to break the code. (Some investigations in that direction can be found in *Irving* [1979], *Knuth* [1976–77] and *Viaud* [1979].) Needless to say, one does not need probability assumptions for that problem.

When we are analyzing a game with n possible colours and k possible positions the set of all possible codes is the set of all k-permutations of n numbers with or without repetitions, respectively. The number of all possible codes therefore is n^k for games with possible repetitions of colours and $\binom{n}{k} k!$ for games with no repetitions allowed.

In the rest of this paper, k will always equal 4, and n will be a number between 4 and 7. (It was not possible to let the programs run for larger problems for computer time reasons.)

The following table gives the number of possible codes for all versions of the game studied in this paper.

As already mentioned, the number of positions will always be 4.

Hence, the game is fully described if the number of colours is given and if it is stated whether or not repetitions of colours are allowed.

Game version	Abbrev.	Number of possible codes
4 colours, no repetitions	4N	24
4 colours, repetitions allowed	4R	256
5 colours, no repetitions	5N	120
5 colours, repetitions allowed	5R	625
6 colours, no repetitions	6N	360
6 colours, with repetitions	6R	1296
7 colours, no repetitions	7N	840
7 colours, repetitions allowed	7R	2401

Tab. 1

Furthermore, the following notations will be used throughout the rest of the paper: Let C_0 denote the set of all possible codes for a certain version of the game (for the number of elements of C_0, see Table 1).

The symbol q (possibly with an index) will denote a question posed by the player. Trivially, $q \in C_0$ holds.

Let A be the set of possible answers to the questions. As mentioned above, every answer can be represented by a pair of integers, the first giving the number of correct colours in correct positions and the second giving the number of correct colours in wrong positions. Equivalent to this form of answer is another one used by the algorithms described in this paper: The first integer a (1) gives the number of colours of the secret code which also occur in the question. The second number a (2) gives the

number of matching colours and positions between the secret code and the questions.

So if $c = (c(1), c(2), c(3), c(4))$ is some secret code and $q = (q(1), q(2), q(3), q(4))$ the numbers $a(1)$ and $a(2)$ may be calculated in the following way [as already stated in *Irving*].

Let v denote a vector whose number of components n is the number of colours allowed in the game considered:
If $v_c(i)$ denotes how many times the colour i occurs in c and $v_q(i)$ denotes the same number for q then

$$a(1) = \sum_{i=1}^{n} \min(v_c(i), v_q(i)).$$

(If the same colour occurs k times in s and l times in q then of course we have $\min(k, l)$ matchings of that colour if we disregard positions.)

The calculation of $a(2)$ is more straightforward, $a(2)$ is just the number of times $c(i)$ equals $q(i)$.

Concerning our answers $a = (a(1), a(2))$ the following facts are obvious:

$a(1) \geqslant a(2)$ for all possible answers.

(4, 3) which is numerically possible cannot occur
(if all colours are correct and 3 out of 4 are in the right position the last one cannot be in the wrong position).

In some games, also other answers are impossible: For example, in 4N $a(1) = 4$ always holds.

The end of the game is reached if $a = (4, 4)$.

Now let $a(c, q)$ denote the answer to be given according to the rules above if the secret code is c and the question is q.

We now define sets of codes compatible with a given sequence of questions q_i and answers a_i

$$C_k(q_1, a_1, q_2, a_2, \ldots, q_k, a_k) = \{c \in C_0 / a(c, q_i) = a_i \text{ for } 1 \leqslant i \leqslant k\}.$$

Trivially,

$$C_k(q_1, a_1, \ldots, q_k, a_k) = \bigcap_{i=1}^{k} C_1(q_i, a_i).$$

Immediate consequences of this equation are

$$C_{k+1}(q_1, a_1, \ldots, q_k, a_2, q_{k+1}, a_{k+1}) = C_k(q_1, a_1, \ldots, q_k, a_k) \ldots \cap C_1(q_{k+1}, a_{k+1})$$

and therefore also

$$C_k(q_1, a_1, \ldots, q_k, a_k) = C_k(q_{\pi(1)}, a_{\pi(1)}, \ldots, q_{\pi(k)}, a_{\pi(k)})$$

for every permutation π of the numbers from 1 to k.

The second of these three equations can be used to calculate the sets C_k stepwise (for a fixed sequence of questions and answers).

Now let us define "strategy":

A strategy is a prescription which questions to pose next for a given sequence of questions and answers posed and received up to the very moment. So every strategy can be formulated in form of a family of functions:

$f_1 = q$ (The first question has no questions before it, so it is fixed)

$f_{k+1}(q, a_1, q_2, a_2, \ldots, q_k, a_k)$ denotes the question to be posed next if the first k questions and answers were q_i and a_i respectively.

Of course

$f_{k+1}(q, a_1, \ldots, q_k, a_k)$ has only to be defined if

$a_k \neq (4, 4)$ because the game is finished if $a_k = (4, 4)$.

It also only has to be defined if

$$C_k(q, a_1, \ldots, q_k, a_k) \neq \emptyset .$$

$C_k = \emptyset$ never can happen during a real game, because the secret code s always must belong to C_k.

(During a real game, of course $a_k = a(s, q_k)$ for a fixed s.)

f_{k+1} has only to be defined for sequences $(q_1, a_1, \ldots, q_k, a_k)$ where $q_{i+1} = f_{i+1}(q, a_1, \ldots, q_i, a_i)$.

So the domain of f_{k+1} depends on domain and range of f_k.

One more concept is important before we begin to calculate strategies:

The concept of equivalence of codes and strategies, respectively. Two codes c and d are considered to be equivalent ($c \sim d$) if there exists a permutation of the numbers 1 to n (where n is the number of possible colours) and a permutation for the numbers of the positions (1 to 4 in all our calculated cases) such that permuting the numbers representing the colours and the numbers of the positions by the respective permutations transforms code c into code d.

((1, 1, 2, 3) can be transformed into (2, 1, 2, 5) by

permutation $\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 5 & 4 & 3 \end{pmatrix}$ for the colours and

permutation $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 1 & 3 & 2 & 4 \end{pmatrix}$ for the positions.

(1, 1, 2, 3) cannot be transformed into (1, 2, 3, 4)).

For two equivalent codes the transformation-generating permutations need not be unique. ($\begin{pmatrix} 1 & 2 & 3 & 4 & 5 \\ 2 & 1 & 5 & 4 & 3 \end{pmatrix}$ and $\begin{pmatrix} 1 & 2 & 3 & 4 \\ 3 & 1 & 2 & 4 \end{pmatrix}$ are another possibility of transforming (1, 1, 2, 3) into (2, 1, 2, 5)).

The useful property of equivalence of codes is the following: If $c_1 = t(c)$ and $q_1 = t(q)$ with a fixed equivalence transformation t (i.e. position- and colour-permutations are the same), then $a(c_1, q_1) = a(c, q)$.

Two strategies f and g are considered to be equivalent if there exists one fixed transformation t of codes of the type considered above such that

$$g_{k+1}(t(q_1), a_1, t(q_2), a_2, \ldots, t(q_k), a_k) = \\ = t(f_{k+1}(q_1, a_1, q_2, a_2, \ldots, q_k, a_k)).$$

This means that in the list describing f one has to replace every code c by $t(c)$ to create the list for g.

Straightforward considerations show that equivalent strategies have equal probabilities for all possible numbers of questions needed to break the code if the probabilities for all possible codes are equal. Hence, the expectations for the number of questions needed are equal too.

This concept of equivalence has an important application. If one has found a strategy f with first question $f_0 = c$ and $c \sim d$ one can easily construct a strategy g equivalent to f with first question d just using one of the transformations transforming c into d for the total strategy f. So when searching for good strategies among certain classes one has only to search for good strategies with only one first question out of every equivalence class of codes. In games without repetitions of colours all codes are equivalent trivially and therefore it is sufficient to consider a set of strategies with a fixed first question.

Before we start with trying to find some good strategies we will derive lower bounds for expectation and maximum number of questions of a strategy.

As noted in *Viaud* [1979] and implicitly used in *Irving* [1979] one easily can see the following:
If we have a set M of m remaining secret codes, a is the number of elements of A (i.e. all possible answers for a certain game), then it is impossible to find a strategy which will break the code with less then p questions where

$$\frac{(a-1)^p - 1}{a-2} < m.$$

(One can only identify 1 code with 1 question, $(a-1)$ codes with 2 questions, $(a-1)^2$ codes with 3 questions and so on.)

So starting on M for every strategy the maximum number of questions needed will be greater than

$$l(M) = p \quad \text{where } \frac{(a-1)^p - 1}{a-2} < m \leqslant \frac{(a-1)^{p+1} - 1}{a-2}.$$

Slightly modifying this reasoning yields that the best expectation one can possibly reach with m secret codes is

$$e(M) = \sum_{i=1}^{p-1} i \cdot (a-1)^i + \left(m - \frac{(a-1)^p - 1}{a-2} \right) p$$

where

$$\frac{(a-1)^p-1}{a-2} < m \leqslant \frac{(a-1)^{p+1}-1}{a-2}$$

and

$$n(M) > 1.$$

($n(M)$ denotes the number of elements of the set M.)

With these results, one can calculate lower bounds for minimax and expectation-minimizing strategies.

One also can go one step further:
We do not use these formulas for the set C_0 itself but rather we shall use them for the sets $C_1(q, a)$ for every question q separately for all $a \in A$.

Defining

$$l_1(q) = 1 + \max_{a \in A} (l(C_1(q, a)))$$

$$e_1(q) = 1 + \frac{1}{n(C_0)} \sum_{\substack{a \in A \\ a \neq (4,4)}} n(C_1(q, a))\, e(C_1(q, a))$$

we find that no strategy can perform better than

$$\min_{q \in C_0} l_1(q) \quad \text{for maximal}$$

and

$$\min_{q \in C_0} e_1(q) \quad \text{for expected number of questions needed.}$$

For our games these formulas yield the following numbers:

		Bounds derived from C_0		Bounds derived from $C_1(q, a)$	
Game	$n(A)$	Expectation	Maximum	Expectation	Maximum
4N	4	3.25	4	3.25	4
4R	14	3.2266	4	3.4063	4
5N	8	3.45	4	3.5000	4
5R	14	3.6832	4	3.7328	4
6N	11	3.6583	4	3.6944	4
6R	14	3.8472	4	4.0093	5
7N	13	3.7964	4	3.9299	5
7R	14	3.9263	5	4.3344	5

Tab. 2

(Naturally one does not need the bounds derived from C_0 if one knows those from C_1, but this table illustrates the differences in the bounds by using a simple and a slightly more sophisticated algorithm.)

2. Different Strategies

2.1 Normal Strategy

The first idea one gets for a strategy is the following:
For a given sequence of questions and answers $q_1, a_1, \ldots, q_k, a_k$ take for $f_{k+1}(q_1, a_1, \ldots, q_k, a_k)$ the lexicographically first element for $C_k(q_1, a_1, \ldots, q_k, a_k)$ i.e. that element $e = (e(1), \ldots, e(4))$ for which $o(e) = \sum_{i=1}^{4} e(i)$. n^{4-i} is the smallest of all such numbers for all elements of $C_k(q_1, a_1, \ldots, q_k, a_k)$ (n is the number of possible colours)

$$f_{k+1}(q_1, a_1, \ldots, q_k, a_k) := c \in C_k(q_1, a_1, \ldots, q_k, a_k) \text{ with } o(c) =$$

$$= \min \{o(d), d \in C_k(q_1, a_1, \ldots, q_k, a_k)\}.$$

This strategy simulates a player who just does not make mistakes, i.e. who for a given set of questions and answers is able to find an element in C_0 which is in accordance with all these questions and answers.

Now let us have a look at this strategy in case of games without repetitions. In this case, we could slightly alter the strategy by using another first question than the lexicographically first one (1, 2, 3, 4). This could possibly have an effect on the expectation because the resulting strategy generally will not be equivalent to that with (1, 2, 3, 4) as the first question because the equivalence transformations between strategies do not preserve the lexicographical order. In spite of this fact the calculations were only done for a first question (1, 2, 3, 4) because all other first questions are equivalent to it and this strategy was not calculated for optimization purposes but as to serve as a base of comparison for more refined strategies.

The situation is somewhat different for games with repetitions. In this case, the calculations were done for the lexicographically first question out of every equivalence class of codes.

Type of game	Expectation	Maximal length	First question
4N	3.667	6	
4R	3.953	6	1234
5N	3.975	6	
5R	4.499	7	1123
6N	4.147	6	
6R	5.147	9	1234
7N	4.517	7	
7R	5.692	10	1234

Tab. 3

In the sequel, we shall call this sort of strategy normal strategy.

(For the game 4R for example these codes are (1, 1, 1, 1), (1, 1, 1, 2), (1, 1, 2, 2), (1, 1, 2, 3) and (1, 2, 3, 4)).

For the game 5R the best strategy of the considered type did not have (1, 2, 3, 4) as the first question but (1, 1, 2, 3) instead.

The results for the expectations and maximums of numbers of questions needed for these strategies are given in Table 3.

2.2 One-Step-Ahead Expectation Minimizing-Strategies

The next idea one might get to find good strategies is the following:
For a given set C of possible codes every question q generates a partition $P(q, C)$ of C with

$$P(q, C) = \{D(q, a)/D(q, a) = C \cap C_1\ (q, a) \text{ for some } a \in A \text{ and } D(q, a) \neq \emptyset\}. \tag{2.2.1}$$

If one has a rough estimate est for the expected number of questions needed for all the $D(q, a)$, then one can calculate an estimate for C (starting with question q) by the formula

$$es(q, C) = 1 + \frac{1}{n(C)} \sum_{D(q,a) \in P(q,C)} n(D(q, a)) \cdot \text{est}(D(q, a)). \tag{2.2.2}$$

We tried three different estimates for est, namely

$e_1(M) = \log(n(M))$

$e_2(M) = n(M)$

$e_3(M) = e(M)$ as defined in part 1 of the paper (i.e. the lower bound for the expectation).

Using e_1 has another interpretation:
$P(q, C)$ is a partition of C_0. $-(es(q, C) - 1)$ then is exactly the entropy of this partition, where entropy is a measure of "goodness" of a partition in sense of information contained in that partition (e.g. subdividing a partition has the effect of increasing entropy. For a detailed discussion of entropy see *Kullback* [1959] e.g.). So in this case by minimizing (2.2.2) we search for the partition with maximal entropy.

Taking e_2 is a rather pessimistic view because $(e_2(M))/2$ is the expected number of questions if there are only questions which yield one answer for one code and a second answer for all the other codes in the set. Minimizing expectation with e_2 is equivalent to minimizing the second order measure of information for $P(q, C)$ over q. (For this measure of information see *Renyi* [1973] e.g..) This function is used in *Irving* [1979] but with a somewhat different interpretation. No such underlying general principle can be attributed to e_3 because e_3 is designed for our special problem.

Now the algorithm has to minimize $es(q, C)$ over some set of questions q and to select a question yielding the minimal $es(q, c)$.

The first possibility is to use only questions out of C and if there are questions with the same value of $es\,(q, C)$ we take the lexicographically first one.

So in slightly abbreviated notation our algorithm reads as follows:

$$
\begin{aligned}
&f_{k+1}\,(q_1, a_1, \ldots, q_k, a_k) := c \in D\,(q_1, a_1, \ldots, q_k, a_k)\\
&\text{with } o\,(c) = \min_{d \in D}\,\{o\,(d)\} \text{ where}\\
&D\,(q_1, a_1, \ldots, q_k, a_k) := \{d \in C_k\,(q_1, a_1, \ldots, q_k, a_k)\,/\\
&es\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k)) =\\
&\qquad \min_{e \in C_k(q_1, a_1, \ldots, g_k, a_k)} \{es\,(e, C_k\,(q_1, a_1, \ldots, q_k, a_k))\}\}.
\end{aligned}
\tag{2.2.3}
$$

Because of the equivalence considerations of section 1 these calculations only were done for each lexicographically first code out of every equivalence class for finding the initial question.

The second possibility is to use all $q \in C_0$ for minimizing $es\,(q, C)$. In this case a proposal of Knuth has proved extremely useful. When searching for the next question when C is the remaining set of codes we use a question $q \in C_0 \backslash C$ only if it is strictly better than all $q' \in C$. So our algorithm in this case is the following:

$$
\begin{aligned}
&f_{k+1}\,(q_1, a_1, \ldots, q_k, a_k) := c \in E\,(q_1, a_1, \ldots, q_k, a_k)\\
&\text{with } o\,(c) = \min_{d \in E}\,\{o\,(d)\} \text{ where}\\
&E\,(q_1, a_1, \ldots, q_k, a_k) := \{d \in C_k\,(q_1, a_1, \ldots, q_k, a_k)\,/\\
&\qquad es\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k)) = \min_{e \in C_0} es\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k))\}
\end{aligned}
\tag{2.2.4}
$$

if

$$
\min_{e \in C_0} es\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k)) =
\min_{e \in C_k(q_1, a_1, \ldots, q_k, a_k)} es\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k))
$$

and

$$
\begin{aligned}
&E\,(q_1, a_1, \ldots, q_k, a_k) := \{d \in C_0 / es\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k)) =\\
&\qquad \min_{e \in C_0} es\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k))\}
\end{aligned}
$$

if

$$
\min_{e \in C_0} es\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k)) <
\min_{e \in C_k(q_1, a_1, \ldots, q_k, a_k)} es\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k)).
$$

Of course it cannot be seen just from the definitions (2.2.3) and (2.2.4) which strategy produces better expectations. However, calculations with e_1 have shown that using definition (2.2.3) produced better results than using definition (2.2.4) only for the game 4R (expectations 3.5547 and 3.5625 respectively), but at the cost of a greater maximum (5 and 4 respectively). Therefore and for reasons of computer time calculations for e_2 and e_3 were done only according to (2.2.4).

The results can be seen in the following table:

	Strategies found by minimizing					
	e_1		e_2		e_3	
Type of game	Expectation	Maximum	Expectation	Maximum	Expectation	Maximum
4N	3.5833	5	3.5833	5	3.5833	6
4R	3.5625	4	3.5871	4	3.5547	5
	(1123)		(1123)		(1123)	
5N	3.8583	5	3.8583	5	3.8667	5
5R	3.9616	5	3.9760	5	3.9616	5
	(1123)		(1123)		(1123)	
6N	4.0278	5	4.0528	6	4.0278	6
6R	4.4151	6	4.3951	6	4.3943	6
	(1234)		(1123)		(1123)	
7N	4.3536	6	4.3619	6	4.3714	6
7R	4.7388	6	4.7101	6	4.7568	6
	(1234)		(1234)		(1234)	

Tab. 4

(For games with repetitions the lines following the expectations and maxima give the first question for the respective strategy.) This table shows that no one of our three types of strategies is strictly better than the other ones, in the sense of performing better for every game.

2.3 Minimax Strategies

If our aim is to minimize the maximal number of questions instead of the expectation we have to modify (2.2.2), (2.2.3) and (2.2.4). Now est (C) has to be an estimate for the maximal number of questions starting from set C and

$$es\,(q, C) := 1 + \max_{D(q,a)\in P(q,C)} (\text{est}\,(D\,(q, a))). \tag{2.3.1}$$

Since every reasonable estimate of the maximal number of questions needed for a set (in one fixed type of game) should be a nondecreasing function of the number of elements of that set all these algorithms are equivalent to the algorithm with

est $(C) = n\,(C)$. So one arrives at the algorithm which was used by *Knuth* [1976–77] for 6R and independently by *Viaud* [1979] for 4R and 6R and which in our notation looks like this:

With

$$le\,(q, C) = \max_{D(q,a) \in P(q,C)} \{n\,(D\,(q, a))\}$$

(where $D\,(q, a)$ and $P\,(q, C)$ are defined according to (2.2.1))

$$f_{k+1}\,(q_1, a_1, \ldots, q_k, a_k) := c \in F\,(q_1, a_1, \ldots, q_k, a_k)$$

with

$$o\,(c) = \min_{d \in F(q_1, a_1, \ldots, q_k, a_k)} \{o\,(d)\}$$

where

$$F\,(q_1, a_1, \ldots, q_k, a_k) = \{d \in C_k\,(q_1, a_1, \ldots, q_k, a_k) \,/\, le\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k)) = \min_{a \in C_0} \{le\,(q, C_k\,(q_1, a_1, \ldots, q_k, a_k))\}\}$$

if

$$\min_{e \in C_0} le\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k)) = \min_{e \in C_k(q_1, a_1, \ldots, q_k, a_k)} le\,(e\,(C_k\,(q_1, a_1, \ldots, q_k, a_k))$$

and

$$F\,(q_1, a_1, \ldots, q_k, a_k) = \{d \in C_0 \,/\, le\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k) = \min_{e \in C_0} \{le\,(q, C_k\,(q_1, a_1, \ldots, q_k, a_k))\}\}$$

if

$$\min_{e \in C_0} le\,(d, C_k\,(q_1, a_1, \ldots, q_k, a_k)) < \min_{e \in C_k(q_1, a_1, \ldots, q_k, a_k)} le\,(e, C_k\,(q_1, a_1, \ldots, q_k, a_k)).$$

The results of this strategy are:

Type of game	Expectation	Maximum	
4N	3.8753	5	
4R	3.5898	4	(1123)
5N	3.9750	5	
5R	4.1008	5	(1122)
6N	4.1389	5	
6R	4.4761	5	(1122)
7N	4.4226	6	
7R	4.8367	6	(1234)

Tab. 5

(As in table 4, the first question to be posed in games with repetitions has been noted.)

Comparisons with table 4 show that the minimax strategy is better than all our three expectation-minimizing strategies in the sense of a smaller maximal number of questions only for 6R.

Unsurprisingly this sort of strategy yields a higher expectation than the strategies of chapter 2.2 for each of our games.

One may also notice that only for the games 4R and 6R this strategy reaches the theoretical bounds of Table 2 for the maximum.

2.4 Optimizing Strategies

The aim of this paper is to find strategies with small expectations for the number of questions needed. In the three types of strategy considered until now this expectation never entered the calculations directly. A strategy was constructed according to some rule and the expectation was calculated a posteriori. But it is also possible to construct a strategy which uses the criterion of minimizing the expectation as the construction rule.

The method is roughly the following:
First, we assume that the best strategies and their expectations for small subsets of C_0 are known. For a given larger set C we now try to calculate the best strategy and its expectation in the following way: Every possible question q generates a partition of C. If the best strategy and its expectation for all sets out of this partition are known we can calculate the best possible strategy and its expectation beginning with question q when C is the set of possible codes in the following way.

Let n_i be the number of elements in the sets C_i of the partition of C and n be the number of elements in C.

Let $p_i = n_i/n$.

Let $E(C_i)$ be the best possible expectation for the set C_i then the best expectation under the conditions just mentioned is $E(q, C) = 1 + \sum_i p_i E(C_i)$.

Because the probabilities for all codes are equal we get an element of C_i with probability p_i when posing question q. So we have to take the weighted mean of the expectations of C_i and we have to add 1 for posing question q. This line of argument can be formalized and leads to a problem of dynamic optimization with the function to be optimized defined recursively. (For a detailed description of this method see *White* [1969] e.g.)

Now let us state the construction of such strategies in detail:
We define two sets of functions recursively.

Let $q_1, a_1, \ldots, q_k, a_k$ be a sequence of questions and answers such that $q_i \neq q_j$ for $i \neq j$ and $C_k(q_1, a_1, \ldots, q_k, a_k) \neq \emptyset$. We define $E(C_k(q_1, a_1, \ldots, q_k, a_k))$ as the best possible expectation for the number of questions needed to finish the game if we have played the sequence $q_1, a_1, \ldots, q_k, a_k$ up to the very moment; furthermore, we define $g_k(q_1, a_1, \ldots, q_k, a_k)$ as one of the questions which must be posed to

play the game in the (statistically) best possible way from that moment on. So we arrive at the following definitions:

1) If $a_k = (4, 4)$ and therefore $n(C_k(q_1, \ldots, a_k)) = 1$

$$E(C_k(q_1, a_1, \ldots, q_k, a_k)) := 0 \tag{2.4.1}$$

g_k need not be defined.
These definitions are totally trivial; in this situation the game is finished.

2) If $n(C_k(q_1, a_1, \ldots, q_k, a_k)) = 1$ and $a_k \neq (4, 4)$

$$E(C_k(q_1, a_1, \ldots, q_k, a_k)) := 1 \tag{2.4.2}$$

$$g_k(q_1, a_1, \ldots, q_k, a_k) := c \text{ where } c \in C_k(q_1, a_1, \ldots, q_k, a_k).$$

In this situation only one possibility is left for the secret code so this is the next question to be posed.

3) If $n(C_k(q_1, a_1, \ldots, q_k, a_k)) = 2$

$$E(C_k(q_1, a_1, \ldots, q_k, a_k)) := 1{,}5 \tag{2.4.3}$$

$$g_k(q_1, a_1, \ldots, q_k, a_k) := c \quad \text{where } c \in C_k(q_1, a_1, \ldots, q_k, a_k)$$

$$\text{and} \quad o(c) = \min_{s \in C_k(q_1, a_1, \ldots, q_k, a_k)} (o(s)).$$

In this case two codes are possible. The probability for each code is 0,5. If our question is one of the two codes we finish the game with probability 0,5 and we need one more question with probability 0,5. Every other question will need at least one more question in every case so the expectation in these cases would be 2 at least.

4) If $n(C_k(q_1, a_1, \ldots, q_k, a_k)) > 2$

we define the following additional functions (A is the set of possible answers)

$$E_1(q, C_k(q_1, a_1, \ldots, q_k, a_k)) := 1 +$$

$$+ \sum_{a^j \in A} p_j E(C_{k+1}(q_1, a_1, \ldots, q_k, a_k, q, a^j)) \tag{2.4.4}$$

$$C_{k+1}(q_1, a_1, \ldots, q_k, a_k, q, a^j) \neq \emptyset$$

where $p_j = n(C_{k+1}(q_1, a_1, \ldots, q_k, a_k, q, a^j)) / n(C_k(q_1, a_1, \ldots, q_k, a_k))$.

(The family of sets $C_k(q_1, a_1, \ldots, q_k, a_k, q, a^j)$ obviously forms a partition of $C_k(q_1, a_1, \ldots, q_k, a_k)$ and is induced by the different answers of the codes to the fixed question q.)

Now we define

$$E(C_k(q_1, a_1, \ldots, q_k, a_k)) := \min_{q \in C_0 \setminus \{q_1, \ldots, q_k\}} E_1(q, C_k(q_1, a_1, \ldots, q_k, a_k)) \tag{2.4.5}$$

$$g_k(q_1, a_1, \ldots, q_k, a_k) := c \text{ where } c \in M \text{ and } o(c) = \min_{d \in M}(o(d)) \text{ with}$$

$$M = \{d \in C_0 \setminus \{q_1, \ldots, q_k\} / E(C_k(q_1, \ldots, q_k)) = \tag{2.4.6}$$

$$= E_1(d, C_k(q_1, \ldots, a_k))\}.$$

This recursive definition is possible because if we pose enough questions every set C_k splits into sets of only one element. (Remark: Case 3) is a consequence of case 4) and may be omitted. Nevertheless, in a computer program the explicit statement of this part of the definition saves computing time.)

5) Definition (2.4.6) can be modified slightly:
We define additional functions
$L(C_k(q_1, a_1, \ldots, q_k, a_k))$ in the following way:

$$L(C_k(q_1, a_1, \ldots, q_k, a_k)) := E(C_k(q_1, a_1, \ldots, q_k, a_k)) \tag{2.4.7}$$

and

$$h_k(q_1, a_1, \ldots, q_k, a_k) := g_k(q_1, a_1, \ldots, g_k, a_k)$$

if $n(C_k(q_1, a_1, \ldots, q_k, a_k)) = 1$

for the further defining equalities we need an additional function (in analogy to the definitions of E).

$$L_1(q, C_k(q_1, a_1, \ldots, q_k, a_k)) = 1 + \max_{a^j \in A} L(C_{k+1}(q_1, \ldots, a_k, q, a^j))$$

$$C_{k+1}(q_1, \ldots, a_k, q, a^j) \neq \emptyset$$

(defined in analogy to E_1)

$$L(C_k(q_1, a_1, \ldots, q_k, a_k)) = \min_{q \in M} L_1(q, C_k(q_1, a_1, \ldots, q_k, a_k)). \tag{2.4.8}$$

Now we define

$$M_1 = \{d \in M / L(C_k(q_1, a_1, \ldots, q_k, a_k)) = L_1(d, C_k(q_1, a_1, \ldots, q_k, a_k))\}$$

$$h_k(q_1, a_1, \ldots, q_k, a_k) := c \text{ where } c \in M_1 \text{ with } o(c) = \min_{d \in M_1}(o(d)).$$

With these definitions L is the maximal number of questions needed completing a sequence $q_1, a_1, \ldots, q_k, a_k$ if our primary aim is to minimize the expectation of the number of questions and our second aim is to choose a question which minimizes the maximum number of questions out of those which minimize expectation. The strategies g_k and h_k yield the same result for the expectation but h_k may perform better for the maximal length of the game.

6) These definitions can be altered in another way:
We may want strategies with $q_{k+1} \in C_k(q_1, a_1, \ldots, q_k, a_k)$ i.e. we only pose questions belonging to the set of codes compatible with all questions and answers given until now. (Due to limited computer time available programs could only be run under these restrictions.)
With these constraints we arrive at the following definitions:
For every sequence $q_1, a_1, \ldots, q_k, a_k$ with $q_{i+1} \in C_i(q_1, a_1, \ldots, q_i, a_i)$ for $0 \leqslant i \leqslant k-1$ we define

α) If $a_k = (4,4)$ and therefore $n(C_k(q_1, a_1, \ldots, q_k, a_k)) = 1$ then

$$E(C_k(q_1, a_1, \ldots, q_k, a_k)) := 0$$

$$L(C_k(q_1, a_1, \ldots, q_k, a_k)) := 0$$

h_k need not be defined.

β) If $n(C_k(q_1, a_1, \ldots, q_k, a_k)) = 1$ and $a_k \neq (4,4)$ then

$$E(C_k(q_1, a_1, \ldots, q_k, a_k)) := 1 \tag{2.4.9}$$

$$L(C_k(q_1, a_1, \ldots, q_k, a_k)) := 1$$

$$h_k(q_1, a_1, \ldots, q_k, a_k) = c \text{ where } c \in C_k(q_1, a_1, \ldots, q_k, a_k).$$

γ) If $n(C_k(q_1, a_1, \ldots, q_k, a_k)) > 1$ then
for every $q \in C_k(q_1, a_1, \ldots, q_k, a_k)$

$$E_1(q, C_k(q_1, a_1, \ldots, q_k, a_k)) := 1 + \sum_{a^j \in A} p_j E(C_{k+1}(q_1, a_1, \ldots, q_k, a_k, q, a^j)) \tag{2.4.10}$$

$$C_{k+1}(q_1, \ldots, a_k, q, a^i) \neq \emptyset$$

where

$$p_j = \frac{n(C_{k+1}(q_1, \ldots, a_k, q, a^j))}{n(C_k(q_1, \ldots, a_k))}$$

and

$$E(C_k(q_1, a_1, \ldots, q_k, a_k)) := \min_{q \in C_k(q_1, \ldots, a_k)} E_1(q, C_k(q_1, a_1, \ldots, q_k, a_k)). \tag{2.4.11}$$

With

$$M(q_1, a_1, \ldots, q_k, a_k) = \{q \in C_k(q_1, \ldots, a_k) / E_1(q, C_k(q_1, \ldots, a_k)) = E(C_k(q_1, \ldots, a_k))\}$$

and

$$L_1(q, C_k(q_1, a_1, \ldots, q_k, a_k)) := 1 + \max_{a^j \in A} L(C_{k+1}(q_1, \ldots, a_k, q, a^j))$$

$$C_{k+1}(q_1, \ldots, a_k, q, a^j) \neq \emptyset$$

and

$$L(C_k(q_1, a_1, \ldots, q_k, a_k)) := \min_{q \in M(q_1, \ldots, a_k)} L_1(q, C_k(q_1, a_1, \ldots, q_k, a_k))$$

and

$$N(q_1, a_1, \ldots, q_k, a_k) = \{q \in M(q_1, a_1, \ldots, q_k, a_k) / L_1(q, C_k(q_1, a_1, \ldots, q_k, a_k)) = L(C_k(q_1, a_1, \ldots, q_k, a_k))\}$$

we define

$$h_k(q_1, a_1, \ldots, q_k, a_k) := c \text{ where } c \in N(q_1, a_1, \ldots, q_k, a_k) \text{ and } o(c) = \min_{d \in N(q_1, \ldots, a_k)} (o(d)).$$

(In analogy to 3) for

$n(C_k(a_1, \ldots, q_k)) = 2$ one has

$E(C_k(a_1, \ldots, q_k)) = 1{,}5$

$L(C_k(a_1, \ldots, q_k)) = 2.$

Since the minimum in (2.4.11) is taken over a smaller set than in (2.4.5) this strategy may give worse results than the previously defined one. But it needs far less computer time and therefore it is possible to do the calculation with these formulas when it cannot be done with the formulas from (2.4.1) to (2.4.6). The strategy given by these functions h_k is optimal in the sense of minimizing the expectation of the number of questions. It also minimizes the maximal number of questions among all strategies minimizing expectation. (There may exist a strategy with smaller maximum but greater expectation.)
Remark: Not all the information contained in the functions h_k is necessary for playing the game. As stated in section 1 a strategy has only to be defined for sequences $q_1, a_1, \ldots, q_k, a_k$ with $q_{j+1} = f_j(q_1, a_1, \ldots, q_j, a_j)$ for $1 \leqslant j \leqslant k-1$. Our domains of h_k are larger, they include all sequences with

$$q_{j+1} \in C_j(q_1, a_1, \ldots, q_j, a_j) \text{ for } 1 \leqslant j \leqslant k-1.$$

Computer runs for this strategy for the games 4N to 6N gave the following results (for reasons of computer time it was impossible to do the calculations for the games 6R, 7N and 7R).

Type of game	Expectation	Maximal length
4N	3.5833	6
4R	3.5469	5
5N	3.8833	6
5R	3.9616	5
6N	4.0361	6

Tab. 6

3. Comparison of the Different Strategies

To state it once more: Our optimizing strategies are not optimal in the set of all strategies in the sense of minimizing the expected number of questions. But they are optimal in the set of all strategies which pose only questions compatible with all questions and answers given up to the moment.

Having a closer look at tables 4, 5 and 6 one might notice the following: If we consider the games with repetitions and the games without separately, the expectation for each strategy increases with the number of possible codes. This is not the case if we consider all the games at once.

The expectations for the game 5N are greater than those for 4R for each type of strategy despite the fact that 4R has more than twice the number of codes than 5N. The reason for this lies in the fact that in the games with repetitions there are more possible answers. For example: In the game 6N the answers (0, 0) (1, 0) (1, 1) are impossible. So a question can create more subsets in a set of possible codes and such can differentiate better.

There is another interesting fact: Compared with the results for games with repetitions the differences between the normal and all the other strategies are rather small for games without repetitions. So one cannot gain very much by calculating sophisticated strategies for these games. This is also in accordance with the author's empirical observation that there are not great differences in the ability of different players for games without repetitions (if one can just find a code compatible with all given questions and answers this can be used as a strategy and for the games considered the expectation of that strategy is always rather near to the expectation of our more sophisticated strategy).

Since it was not possible to do the calculations according to (2.4.1) to (2.4.8) except for 4N where they yielded an expectation of 3.5833 (like all the other strategies) and a maximum of 5 (like some other strategies) we do not have hints about how far we are away from the best expectations we can reach. (Of course we can compare our results with table 2.)

3.1 Compositional Strategies

Irving [1979] tried the following to find a good strategy for the game 6R:
He calculated the first and all second questions according to our algorithm in part 2.2 using e_2: For the resulting smaller sets of codes he essentially used the procedure

described in (2.4.1) to (2.4.8). The result was a strategy with an expectation of 4.3688 and a maximum of 6.

By using the same first question as he does, namely 1123 one can do better by using some other questions, which were found according to the other construction rules of chapter 2.2.

Before we state these situations explicitly we have to take notice of Irvings notation of the answers, which is different from ours:
Irvings notation: $b = (b\,(1), b\,(2))$ where $b\,(1) = a\,(2)$ and $b\,(2) = a\,(1) - a\,(2)$;
$(a = (a\,(1), a\,(2))$ is our notation of chapter 1).

Answer (Irvings notation)	Answer (our notation)	Irvings next question	Our next question	Improvement of expectation	Construction rule
(3, 0)	(3, 3)	6213	1245	0.0015	e_1
(1, 2)	(3, 1)	1432	1234	0.0008	optimized
(0, 3)	(3, 0)	4312	2314	0.0008	e_3
(0, 0)	(0, 0)	4545	4456	0.0015	optimized

Tab. 7

Incidentally, Irvings answer 6213 in the first line is not constructed according to his rule (e_2). The question according to e_2 would be 1233 and would yield a higher expectation.

Also his question 4545 after answer (0, 0) is not the lexicographically first question constructed according to his algorithm which would be 4455.

So by combining the results of different algorithms one can have a strategy with an expectation of 4.3642. This strategy has a maximum of 6 (a detailed description of this strategy can be found in the Appendix).

Since in all parts of that strategy which are different from Irving's strategy we have a maximum of 5 we can use Irving's reasoning and assert that there is only one code to be found with 6 guesses. A slight modification of the strategy yields another strategy with a maximum of 5 and an expectation of 4.3657 which is better than Knuth's strategy with expectation of 4.4761 and Irving's modified strategy with expectation 4.3704 (both of these strategies have a maximum of 5).

Or course all these numbers show that one cannot gain very much by searching which of our algorithms is better, because no one of them is uniformly better than the others, but the main aim of this paper was to compare different rules for constructing a rather good strategy and not only to find one very good strategy.

4. Concluding Remarks

The construction of all our strategies is based on the assumption that the probabilities for all codes of a game are equal. What can be done if this is not the case?

We can calculate our strategies in a modified way for every probability distribution over the codes. We only have to change the weights in (2.2.2), (2.4.4) and (2.4.10) ac-

cording to the probability distribution. So we can compute all types of strategies for these cases. Of course these strategies may have other expectations than those for equal probabilities.

If the game is played against a human codemaker and the strategy is always using the same first question after some time the codemaker will set codes which have shown to need a large number of questions to be guessed. To circumvent such biases the player using our strategies should randomize the first question by choosing a permutation of the numbers representing the colours and another one for the numbers representing the positions at random and then modify our calculated strategies according to the chosen permutation [as proposed in *Knuth*]. In this way we can randomize away preferences of colours and positions. More exactly: This random permutation has the effect that the expectation of the strategy is the same as that of a strategy against a probability distribution over the codes with equal probabilities for all codes of the same equivalence class. So if the probabilities of the equivalence classes are proportional to their sizes this randomizing procedure gives us a strategy with the same expectation as the previously calculated strategy for equal probabilities of all codes. If the probabilities of the equivalence classes are changed there is no such obvious procedure. In this case one would have to know the probabilities for all classes. For the rest one could apply the randomizing procedure. So in games without repetitions (where all codes are equivalent) with our randomized strategy we can reach an expectation equal to that for the game against equal probabilities for all codes for every probability distribution over the codes. This is not the case for games with repetitions.

5. Computation

The strategies were calculated on the CDC CYBER 73 at the Computing Center of the University of Vienna.

Computation for the larger versions of the game was very time-consuming. Therefore the author wishes to thank the staff of the Computing Centre for fulfilling his extensive needs of computing time.

Appendix: Detailed description of the compositional strategy with best expectation for the game 6R. (Modification of Irving's strategy, but also the part's directly adopted from *Irving* [1979] are given)

Our notation is the same as in *Irving* [1979] and therefore we cite its description given in that paper:

„If there are n (>1) codewords remaining at a given stage, then this is represented by

$n\,(y_1\,y_2\,y_3\,y_4)$ if the reply to guess $y_1\,y_2\,y_3\,y_4$ will uniquely determine the codeword, and where the presence of * denotes that $y_1\,y_2\,y_3\,y_4$ could not itself be the codeword,

$n\,(y_1\,y_2\,y_3\,y_4 : \alpha_{04}, \alpha_{03}, \alpha_{02}, \alpha_{01}, \alpha_{00}; \alpha_{13}, \alpha_{12}, \alpha_{11}, \alpha_{10}; \alpha_{21}, \alpha_{21}, \alpha_{20}; \alpha_{30}; \alpha_{40})$ if guess $y_1\,y_2\,y_3\,y_4$ and reply (i, j) lead to situation α_{ij}."

(The reply (i, j) in this description of course is the form $b = (b\,(1), b\,(2))$ of chapter 3.1.)

The strategy:

1296 (1123: 2 (2311), 44*A*, 222*B*, 276*C*, 81*D*; 4 (1312), 84*E*, 230*F*, 182*G*; 5 (1321: 1, 0, 0, 0, 0; 2 (1132), 0, 0, 0; 1, 0, 0; 0; 1), 40*H*, 105*I*; 20*J*; 1)

A = (2314: 2 (3241), 5 (3251: 0, 0, 0, 0, 0; 0, 0, 0, 0; 1, 1, 0; 2 (3231); 1), 0, 0, 0; 2 (2431), 10 (3215: 1, 1, 0, 0, 0; 0, 1, 0, 0; 1, 1, 2 (3411); 2 (3212); 1), 4 (3511), 0; 3 (2341), 8 (3312: 0, 0, 0, 0, 0; 1, 2 (2351), 1, 0; 0, 0, 1; 2 (5312); 1), 6 (2511: 0, 0, 0, 0, 0; 0, 0, 0, 0; 0, 1, 2 (3311); 2 (2211); 0); 3 (2312: 0, 0, 0, 0, 0; 0, 0, 0, 0; 0, 0, 0; 2 (2315); 1); 1)

B = (4532: 4 (2345: 1, 0, 0, 0, 0; 0, 0, 0, 0; 2 (2354), 0, 0; 0; 1), 20 (3256: 1, 1, 4 (2415), 0, 0; 0, 2 (2355), 4 (5214), 0; 2 (3265), 1, 2 (3244); 2 (3255); 1), 43 (3651: 2 (5316), 4 (5216), 5 (5215), 2 (2214), 0; 1, 5 (2615), 7 (1461*), 2 (2241); 2 (6351), 4 (5351), 5 (3535*); 3 (3366*); 1), 14 (3616: 1, 1, 1, 0, 0; 0, 2 (3361), 1, 0; 2 (3661), 1, 2 (5611); 2 (2616); 1), 1; 4 (5342: 1, 0, 0, 0, 0; 2 (3452), 0, 0, 0; 0, 0, 0; 0; 1), 38 (3652: 2 (5236), 5 (2335), 8 (4215: 1, 1, 2 (2334), 0, 0; 1, 0, 1, 0; 0, 0, 0; 1; 1), 0, 0; 2 (5362), 3 (3235), 7 (4351), 0; 1, 3 (3252), 3 (2442*); 3 (3252); 1), 34 (6236: 1, 2 (2362), 4 (5612), 7 (2515), 0; 1, 1, 6 (3516), 5 (4214); 1, 1, 2 (6431); 2 (3236); 1), 12 (6331: 0, 0, 1, 1, 0; 0, 0, 3 (4611), 2 (4411); 1, 0, 1; 2 (3331); 1); 5 (2234: 0, 2 (3542), 0, 0, 0; 0, 1, 0, 0; 0, 1, 0; 1; 0), 14 (2536): 0, 1, 1, 0, 0; 2 (3562), 4 (5232), 2 (4234), 0; 0, 1, 1; 1; 1), 22 (2362: 0, 0, 2 (4631), 3 (3531), 1; 1, 3 (6632), 3 (4212), 3 (4331); 1, 2 (6332), 2 (2512); 1; 0); 10 (6412: 0, 0, 1, 0, 0; 0, 1, 2 (4232), 3 (5222*); 0, 1, 2 (4432); 0; 0); 1)

C = (2445: 1, 12 (4562: 1, 1, 0, 0, 0; 2 (4256), 0, 1, 0; 1, 2 (4252), 2 (4514); 1; 1), 42 (5266: 1, 1, 5 (6514), 6 (3354), 1; 1, 3 (6252), 8 (4516), 5 (5514); 2 (6256), 2 (6264), 3 (4262); 3 (5262); 1), 38 (6561: 1, 3 (3656), 4 (3356), 2 (3634), 1; 0, 5 (5366), 6 (3565*), 2 (3364); 3 (6651), 3 (4661), 5 (6662); 2 (5561); 1), 9 (2363: 0, 1, 1, 1, 0; 0, 2 (3336), 1, 1; 0, 1, 1; 0; 0); 3 (5244), 27 (5246: 3 (2564), 2 (2554), 2 (4451), 0, 0; 3 (4265), 4 (2254), 4 (4541), 0; 1, 2 (6244), 3 (4244); 2 (5242); 1), 51 (4436: 2 (3644), 1, 6 (6242), 6 (2562), 3 (2552); 1, 6 (4365), 6 (6446*), 6 (2566); 2 (4346), 4 (3456), 4 (4535); 3 (4411*); 1), 36 (3646: 0, 2 (6461), 4 (6535), 5 (5535), 1; 1, 2 (6635), 6 (5635), 5 (5655*); 2 (3466), 3 (6641), 3 (5635); 1; 1); 4 (2454), 15 (6245: 1, 2 (2452), 0, 0, 0; 1, 2 (2542), 4 (4415), 0; 0, 1, 2 (4345); 1; 1), 28 (3546: 1, 2 (5435), 5 (2462), 1, 0; 2 (3465), 3 (3435), 4 (2565), 3 (4441); 1, 1, 3 (2246); 2 (3545); 0); 9 (6265: 0, 0, 1, 2 (2444), 0; 0, 1, 2 (2455), 1; 0, 1, 1; 0; 0); 1)

D = (4456: 2 (5644), 8 (5645: 0, 1, 0, 0, 0; 1, 0, 0, 0; 1, 2 (5544), 1; 1; 1), 6 (5545: 0, 0, 0, 1, 0; 0, 0, 0, 1; 0, 0, 2 (5665); 1; 1), 0, 0; 4 (4564), 18 (4565: 1, 2 (5646), 2 (5444), 0, 0; 2 (5546), 2 (5445), 2 (4644), 0; 1, 1, 2 (4544); 2 (4545); 1), 8 (5655), 2 (5555); 5 (4546: 1, 0, 0, 0, 0; 2 (4465), 0, 0, 0; 1, 0, 0; 0; 1), 10 (4566: 0, 1, 1, 0, 0; 0, 2 (4655), 1, 0; 1, 1, 2 (4464); 0; 1), 9 (4555: 0, 0, 0, 1, 0; 0, 0, 1, 3 (4444); 1, 1, 1; 0; 1); 8 (4556: 0, 0, 0, 0, 0; 0, 0, 0, 0; 1, 2 (4455); 3 (4446); 1; 1); 1)

E = (1234: 4 (2413: 2 (3142), 0, 0, 0, 0; 1, 0, 0, 0; 0, 0, 0; 0; 1), 14 (3521: 0, 2 (2313), 1, 0, 0; 2 (2513), 1, 1, 0; 1, 2 (2321), 1; 2 (3321); 1), 13 (2151: 0, 0, 0, 0, 0; 1, 2 (3115), 1, 0; 1, 2 (2112), 2 (3111); 3 (2111); 1), 0, 0; 3 (1342), 18 (2135: 2 (1352), 4 (3213), 2 (1341), 0, 0; 0, 1, 1, 0; 1, 2 (3132), 2 (2114); 2 (2132); 1), 10 (1512: 0, 1, 2 (3131), 0, 0; 0, 1, 1, 0; 0, 1, 2 (1311); 1; 1), 0; 2 (1432), 6 (1314: 0, 0, 0, 0, 0; 1, 1, 2 (1532), 0; 0, 0, 1; 0; 1), 9 (1531: 0, 0, 0, 0, 0; 0, 1, 2 (1212), 0; 0, 1, 2 (1211); 2 (1331); 1); 4 (1232: 0, 0, 0, 0, 0; 0, 0, 0, 0; 0, 0, 1; 2 (1235); 1); 1)

F = (1415: 2 (4151), 12 (2154: 1, 0, 0, 0, 0; 1, 0, 4 (5161), 0; 1, 1, 2 (6151); 1; 1), 34 (6152: 1, 4 (5221), 5 (3524), 0, 0; 1, 3 (2146), 5 (3164), 3 (4134); 2 (5162), 2 (3156), 4 (2142); 3 (5152); 1), 41 (4233: 1, 3 (2324), 4 (3522), 6 (3166), 0; 1, 6 (3224), 4 (2563), 3 (6136); 1, 4 (2533), 4 (5263); 3 (4243); 1), 20 (3626: 1, 3 (6233), 2 (2333), 0, 0; 1, 3 (2633), 2 (2322), 0; 1, 1, 2 (3322); 3 (3622); 1); 2 (1541), 19 (6114: 1, 2 (1561), 4 (1534), 0, 0; 0, 0, 6 (4135), 0; 1, 2 (4111), 1; 1; 1), 39 (1256: 1, 5 (6135), 10 (4335*), 1, 0; 1, 3 (1642), 6 (6616*), 2 (1334); 1, 2 (1552), 3 (1242); 3 (1243*); 1), 23 (6313: 0, 1, 5 (1266), 4 (3422), 0; 1, 1, 3 (2433), 3 (2325); 1, 0, 1; 2 (3313); 1); 3 (4115), 11 (6116: 0, 1, 1, 3 (1452), 0; 0, 1, 1, 1; 0, 1, 1; 1; 0), 17 (1365: 0, 1, 2 (3413), 0, 0; 0, 1, 4 (1611), 1; 1, 1, 2 (1255); 3 (1335); 1); 6 (4614*); 1)

G = (1456: 3 (5164), 14 (6524: 0, 1, 3 (4145), 0, 0; 1, 1, 3 (4164), 0; 1, 1, 2 (6164); 0; 1), 39 (3563: 0, 0, 5 (5622), 7 (4662*), 1; 1, 3 (5343), 4 (6525), 5 (4225*); 2 (5363), 4 (3643), 3 (5543); 3 (5563); 1), 26 (2642: 0, 1, 1, 5 (3363), 3 (3533); 0, 3 (2224), 4 (5222), 6 (3343); 0, 1, 1; 1; 0), 2 (2222); 4 (1554*), 18 (6146: 1, 1, 3 (1565), 0, 0; 0, 1, 7 (1544), 0; 1, 1, 1; 1; 1), 30 (6462: 1, 2 (4226), 3 (2526), 4 (3653), 1; 0, 1, 5 (5424), 4 (6353), 1; 0, 1, 6 (6144*); 2 (6422); 0), 16 (2646: 0, 0, 2 (4424), 2 (3433), 4 (5655*); 0, 1, 2 (2422), 2 (3443); 0, 1, 1; 1; 0); 4 (1665*), 10 (6655: 0, 1, 1, 1, 0; 0, 1, 2 (1554), 1; 0, 1, 1; 1; 0), 9 (6424: 0, 0, 0, 1, 1; 0, 0, 1, 2 (3453); 1, 1, 1; 1; 0); 6 (6655*); 1)

H = (1242: 0, 3 (4121), 8 (3125: 0, 0, 0, 0, 0; 1, 2 (2133), 1, 0; 0, 1, 1; 1; 1), 3 (3513*), 0; 0, 4 (1421), 6 (6334*), 6 (6615: 0, 0, 1, 1, 0; 0, 0, 2 (1135), 1; 0, 0, 1; 0; 0); 0, 2 (1322), 6 (6152*); 2 (1142); 0)

I = (1415: 0, 3 (5124), 18 (4622: 0, 0, 1, 3 (5163), 3 (3153); 0, 2 (2124), 2 (5126), 2 (4133); 0, 1, 3 (5122); 1; 0), 18 (5263: 0, 0, 3 (2126), 1, 0; 1, 3 (2523), 2 (6133), 1; 1, 2 (5323), 3 (3163); 1; 0), 9 (6266: 0, 0, 2 (3623), 2 (2323), 0; 0, 1, 1, 2 (3223); 0, 0, 1; 0; 0); 1, 9 (6141: 0, 0, 1, 0, 0; 1, 1, 2 (4125), 0; 1, 1, 1; 1; 0), 17 (1653: 0, 1, 2 (2125), 0, 0; 0, 1, 4 (1161), 1; 1, 1, 2 (1343); 3 (1353); 1), 12 (6226: 0, 0, 1, 2 (1633), 1; 0, 1, 1, 2 (3423); 0, 1, 2 (6423); 1; 0); 1, 4 (1165), 11 (6323: 0, 1, 2 (1433), 1, 1; 0, 1, 2 (1422), 2 (1225); 0, 0, 1; 0; 0); 2 (1425); 0)

J = (1245: 0, 2 (4123), 3 (2116*), 0, 0; 0, 3 (1124), 6 (1162*), 3 (1116*); 0, 1, 2 (1143); 0; 0)

(This strategy differs from that one given in *Irving* [1979] only in the parts A, D, E and J.)

References

Irving, R.W.: Towards an Optimum Mastermind Strategy. To appear in Journal of Recreational Mathematics 11, 1979.

Knuth, D.E.: The Computer as Mastermind. Journal of Recreational Mathematics 9, 1976–77, 1–6.

Kullback, S.: Information theory and statistics. New York 1959.

Renyi, A.: Wahrscheinlichkeitstheorie mit einem Anhang über Informationstheorie. Berlin 1973.

Viaud, D.: Une Formalisation du Jeu de Mastermind. R.A.I.R.O. Recherche operationelle/ Operations Research 13 (3), 1979, 307–321.

White, D.J.: Dynamic Programming. Edinburgh 1969.