In Bulletin of the Faculty of Engineering, Alexandria University, (Alexandria, Egypt), December 1986.

MasterMind
A game of Diagnosis Strategies

Azer Bestavros   Ahmed Belal
Dept. of Computer Science
Alexandria University
Egypt

September 1986

Abstract

The MasterMind game can be thought of as a diagnosis process where the analyst (diagnoser) – by conducting specific tests – is trying to diagnose a hidden pattern (fault). The goal is to locate the pattern in a minimum time. In this paper we introduce the "Dynamic" MasterMind in which the setter (adversary) has the right to change the hidden pattern provided that the new pattern does not conflict with previous responses. Unlike the "Static" MasterMind, the proposed game turns the setter to an active player and eliminates the possibility of breaking the hidden code accidentally.

We start the paper by presenting a formulation, based on information theory, of the MasterMind game. In this formulation the setter is considered to be a zero memory information source producing messages in response to analyst guesses. The amount of information from such messages, based on a one-step lookahead policy, is defined and used by the analyst to minimize the required number of guesses. In this context, two strategies are presented and evaluated. In the first, based mainly on a *MaxMin* criterion, the analyst tries the guess that, in the worst case, gives the maximum possible amount of information. Next, in an attempt to alleviate the effect of the pool structure, an averaging strategy, based on the maximum *Entropy* is considered in which the analyst tries the guess that, in the average, gives the maximum possible amount of information. The more complex L-step lookahead strategies for both the static and dynamic games are left for future work.

1 The Static Mastermind

Rules for the famous Mastermind game are extremely simple. One player, the *setter*, selects a hidden combination of any N digits in the range 1 to R (repeats are allowed). This hidden pattern is called the *code*. The second player, the *analyst*, tries to uncover the code by a series of probes or *guesses*. A guess, like a code, is any pattern of N digits from 1 to R. In response to each guess, the setter must reply by saying how many digits of the probe are *exact*; *i.e.* match the code in both value and relative position, and how many are *included*; *i.e.* are elements of the code but in wrong positions. The round goes on until the analyst breaks the code; *i.e.* has sufficient amount of information to know the code.

1.1 Formulation

The analyst, starting with no information about the code, must consider all the possible R^N patterns. These patterns form the initial solution pool, P_0. After the i^{th} round, $i = 1, 2, \ldots$, the solution pool reduces to $P_i \subset P_{i-1}$. Obviously, the i^{th} guess is selected from that pool. Let $\{G_i^1, G_i^2, \ldots, G_i^k, \ldots\} = P_{i-1}$ denote the set of possible i^{th} guesses. If guess G_i^k is proposed at the i^{th} round, the new pool P_i consists of all those members of pool P_{i-1} which have not been eliminated by the i^{th} response from the setter. That is, pool P_i is the set of all patterns that might still be the code. Therefore, the goal of the analyst is to reduce some pool, say P_m, to a single element and the game ends when such a pool is found. In this case, we say that the game has ended after m rounds.

1.2 Optimal Guess and L-step lookahead

At any step, i, the analyst's goal is to minimize the number of guesses , m, required to break the code. This, however, does not imply that the optimum guess is that yielding a new pool of minimum size. As a matter of fact, not only the size of this pool but also its structure (*i.e.* the relative correlation of its members which can be estimated from the pool sizes further steps ahead) must be considered. Therefore, an L-step lookahead strategy should select the guess that optimizes the pool sizes L-step further. For example, a two-step lookahead strategy should select the guess G_i^k that optimizes the pool sizes of the $(i+2)^{th}$ step. It is to be noted that, in choosing the value of L (the lookahead level), the analyst is actually determining the number of levels to be considered from a large tree describing the course of the game for any guess and for any possible response. In the rest of this paper we adopt a one-step lookahead policy. Strategies devised and results obtained may be easily extended to L-step lookahead policies.

1.3 Information contents

After the i^{th} round of the game, the 3-tuples (P_{i-1}, G_i^k, R_i^j) may be viewed as a message m_i from an information source, $\mathcal{S}$. The amount of information [1]–[2], that the analyst gets from such messages, namely $I(m_i)$, may be used to get from P_{i-1} a new pool P_i. The greater $I(m_i)$ the smaller the size of the new pool will be. Therefore, in choosing guess G_i^k, the analyst must try to maximize $I(m_i)$. Since P_{i-1} was previously computed in the $(i-1)^{th}$ round, and since guess G_i^k will be uniquely specified in the i^{th} round, the set of possible messages from $\mathcal{S}$ is precisely the set of possible responses $\{R_i^1, R_i^2, \ldots, R_i^j, \ldots\}$ from the setter. Let R_i^j be the setter's i^{th} response. The amount of information from $m_i = (P_{i-1}, G_i^k, R_i^j)$ is:

$$I(m_i) = -\log_2(Prob(R_i^j/(P_i, G_i^k))) \text{ bit(s)}.$$

To calculate $I(m_i)$, we must find the value of $Prob(R_i/P_i^j, G_i^k)$. Any guess G_i^k imposes a partition on the set of patterns in P_{i-1}. Each class in this partition corresponds to a possible response, R_i^j, from the setter. That is, for a guess G_i^k the pool P_{i-1} may be expressed as the union of mutually disjoint sets, or classes, as follows:-

$$P_{i-1} = C_i^k(R_i^1) \cup C_i^k(R_i^2) \cup \ldots \cup C_i^k(R_i^j) \cup \ldots$$

where $C_i^k(R_i^j)$ is simply the set of patterns in P_{i-1} resulting in a response R_i^j when compared to G_i^k. Thus, $C_i^k(R_i^j)$ represents the next pool P_i if R_i^j is the response to G_i^k. Therefore, assuming that each pattern in P_{i-1} is equally likely to be the code, the required probability is simply the ratio of the number of patterns in $C_i^k(R_i^j)$ to the total number of patterns in P_{i-1}. Hence:

$$Prob(m_i) = Prob(R_i^j/(P_{i-1}, G_i^k)) = \tfrac{|C_i^k(R_i^j)|}{|P_{i-1}|}, \text{ and}$$
$$I(m_i) = \log_2(\tfrac{|P_{i-1}|}{|C_i^k(R_i^j)|}) \text{ bit(s)}.$$

It is to be noted that whichever guess is selected from the current pool, a positive amount of information is always gained. Thus, the game always ends after a finite number of guesses, m, and it is the analyst's goal to minimize m. The total amount of information required to identify the hidden pattern is $\log_2(R^N)$ (see Table-1). Therefore the game will end after the m^{th} step, if and only if:

$$\sum_{i=1}^{m} \log_2(I(m_i)) = \log_2(R^N) = N\log_2(R)$$

Two optimal analyst's strategies based on the one-step lookahead policy are singled out and evaluated in the next sections. In the first, based mainly on a MaxMin criterion, the analyst's goal is to maximize the minimum amount of information to be gained from his guess. This strategy is obviously pessimistic. In the second strategy, based on an expected value criterion, the analyst's goal is to maximize the average amount of information (Entropy) to be gained from his guess.

		R								
		2	3	4	5	6	7	8	9	10
	2	2.00	3.17	4.00	4.64	5.17	5.61	6.00	6.34	6.64
	3	3.00	4.75	6.00	6.97	7.75	8.42	9.00	9.51	9.97
N	4	4.00	6.34	8.00	9.29	10.34	11.22	12.00	12.68	13.29
	5	5.00	7.92	10.00	11.61	12.92	14.04	15.00	15.85	16.61
	6	6.00	9.51	12.00	13.93	15.51	16.84	18.00	19.02	19.93

Table 1: Total amount of information (in bits) required to break the hidden code.

1.4 The MaxMin information strategy

In the MaxMin strategy, the analyst selects the guess that, in the worst case, offers the maximum possible amount of information. This is done by computing the minimum information to be gained from every candidate guess, G_i^k. The guess, G_i^o, giving the maximum of these minimums is selected. If more than one guess give this maximum (*i.e.* a tie exists) then the guess giving the maximum of the next-to-minimum's is selected, … *etc.* It is obvious that this strategy is conservative, since it assumes that at each round, and for any selected guess the minimum information will be obtained from the setter's response.[1] Clearly such an assumption is pessimistic and unlikely to hold for all guesses. A complete description of this strategy follows:-

[1]This is possible if the setter maliciously changes his code. We will consider this possibility later on when we discuss the "dynamic" MasterMind.

```
Procedure MaxMin($P_{i-1}$)
  Begin
    - For all possible guesses $G_i^k \in P_{i-1}$
      - For all possible responses $R_i^j$
        - Compute $I(m_i^{k,j})$
      - Sort $I(m_i^{k,j})$ in an ascending order to a vector $V_i^k()$
    - Select guess $G_i^o$ for which $V_i^o(x) \geq V_i^k(x)$, and $V_i^o(y) > V_i^k(y)$ ;
      where $k \neq o$, and $x = 1, 2, \ldots, y-1$.
  End.
```

The above strategy is optimal in the sense that it seeks the minimization of the largest number of guesses needed to find the code by trying to maximize the minimum amount of information from each guess and hence obtaining the smallest possible pool in each step. As a matter of fact, an upper bound on the number of required guesses may be obtained and an analyst using the above algorithm, is guaranteed of breaking the code in a number of guesses less than or equal to this bound.

1.5 The Maximum Entropy Strategy

In the Maximum Entropy strategy, the analyst selects the guess that, in the average, offers the maximum possible amount of information. In other words, the average amount of information (Entropy) from every possible guess G_i^k is computed, and the guess G_i^o giving the maximum of these Entropies is selected. If a tie exists, then the MaxMin strategy may be used to break it. It is obvious that the maximum Entropy strategy is more optimistic than the MaxMin strategy. A complete description of this strategy follows:-

```
Procedure MaxEnt($P_i$)
  Begin
    - For all possible guesses $G_i^k \in P_{i-1}$
      - For all possible responses $R_i^j$
        - Compute $I(m_i^{k,j})$
      - Compute the Entropy $H_k(\mathcal{S})$; where $H_k(\mathcal{S}) = \sum_{j=1}^{r} Prob(m_i^{k,j}) \times I(m_i k, j)$
    - Select guess $G_i^o$ for which $H_o(\mathcal{S}) > H_k(\mathcal{S})$;
      where $k \neq o$.
  End.
```

The above strategy is optimal in the sense that it seeks the minimization of the average number of steps required to break the code by trying to maximize the average amount of information from each guess.

1.6 Initial guess

Of special importance is the initial guess structure, and the amount of information it can provide. In table-2, the optimal initial structures, obtained for different R's and N's, are shown according to both the MaxMin and Maximum Entropy strategies, based on a one-step lookahead policy. Also tabulated, is the amount of information each of these structures provides on the average and in the worst case. The minimum amount of information from the initial guess (corresponding to the worst initial structure) is also shown in Table-2.

1.7 Performance evaluation

Results obtained from a sample of 500 games ($N = 4, R = 6$) played using both the MaxMin and MaxEnt strategies are shown in Table-3. As expected, the average number of steps required to break the code for the MaxMin strategy was 3.860 while it was only 3.835 for the MaxEnt strategy.

1.8 Upper bounds

As discussed earlier, the analyst, whatever his strategy is, will eventually break the code in a finite number of guesses. By constructing the MasterMind tree (showing all possible guesses and responses) the least upper bound for m ($m_{L.U.B.}$) as well as the maximum value for m (M_{max}) can be found. Therefore using any strategy, the upper bound for m, $m_{U.B.}$, must always satisfy the following inequality:-

$$m_{L.U.B.} \leq \ m_{U.B.} \leq \ m_{max}$$

The exact value of $m_{L.U.B.}$, m_{max}, and $m_{U.B.}$ for a certain strategy can be obtained. For example, from the complete MasterMind tree for $N = 2$ and $R = 3$, it was found that $m_{L.U.B.} = 2$ while $m_{max} = 3$; moreover, for both the MaxMin and MaxEnt strategies $m_{U.B.} = 2$.

2 The Dynamic Case

A main disadvantage of the old MasterMind game is the passive nature of the setter. At the beginning of the game, the setter chooses a hidden code and during the rest of the game, his role is just to compare guesses with this "static" code. Another weakness in the game is that luck (not only "Mind" !!) plays a part and it is possible for a lucky analyst to break the code accidentally. In order to overcome these disadvantages, we propose the new "Dynamic" MasterMind, which turns the setter to an active player, and eliminates the role of luck.

Size		Initial Structure			Information Contents in Bits					
N	R	MaxMin	MaxEnt	Worst	MaxMin		MaxEnt		Worst	
					Min.	H(S)	Min.	H(S)	Min.	H(S)
2	2	AA/AB	AA/AB	AA/AB	1.000	1.500	1.000	1.500	1.000	1.500
2	3	AB	AB	AA	1.170	2.059	1.170	2.059	1.170	1.392
2	4	AB	AB	AA	1.415	2.031	1.415	2.031	0.830	1.248
2	5	AB	AB	AA	1.474	1.922	1.474	1.922	0.644	1.124
2	6	AB	AB	AA	1.170	1.803	1.170	1.803	0.526	1.022
2	7	AB	AB	AA	0.971	1.690	0.971	1.690	0.445	0.938
2	8	AB	AB	AA	0.830	1.587	0.830	1.587	0.385	0.868
2	9	AB	AB	AA	0.725	1.495	0.725	1.495	0.340	0.809
2	10	AB	AB	AA	0.644	1.413	0.644	1.413	0.304	0.758
3	2	AAB	AAB	AAA	1.415	2.156	1.415	2.156	1.415	1.811
3	3	AAB	AAB	AAA	2.170	2.763	2.170	2.763	1.170	1.700
3	4	ABC	ABC	AAA	2.093	2.743	2.093	2.743	1.245	1.542
3	5	ABC	ABC	AAA	2.059	2.690	2.059	2.690	0.966	1.405
3	6	ABC	ABC	AAA	1.778	2.593	1.778	2.593	0.789	1.290
3	7	ABC	ABC	AAA	1.667	2.486	1.667	2.486	0.667	1.193
3	8	ABC	ABC	AAA	1.634	2.379	1.634	2.379	0.578	1.111
3	9	ABC	ABC	AAA	1.639	2.277	1.639	2.277	0.510	1.040
3	10	ABC	ABC	AAA	1.544	2.182	1.544	2.182	0.456	0.979
4	2	AAAB	AAAB	AAAA	2.000	2.608	2.000	2.608	1.415	2.031
4	3	AAAB	AABB	AAAA	2.433	3.178	2.340	3.308	1.340	1.920
4	4	AABC	AABC	AAAA	2.476	3.270	2.476	3.270	1.245	1.762
4	5	AABB	AABC	AAAA	2.381	3.070	2.381	3.168	1.288	1.620
4	6	AABB	ABCD	AAAA	2.340	2.885	2.054	3.057	1.052	1.498
4	7	ABCD	ABCD	AAAA	2.045	2.982	2.045	2.982	0.890	1.394
4	8	ABCD	ABCD	AAAA	2.069	2.899	2.069	2.899	0.771	1.305
4	9	ABCD	ABCD	AAAA	1.850	2.812	1.850	2.812	0.680	1.228
4	10	ABCD	ABCD	AAAA	1.714	2.727	1.714	2.727	0.608	1.160

Table 2: Initial guess structure, and the amount of information gained accordingly (in bits).

Size		Initial Structure			Information Contents in Bits					
N	R	MaxMin	MaxEnt	Worst	MaxMin		MaxEnt		Worst	
					Min.	H(S)	Min.	H(S)	Min.	H(S)
5	2	AAAAB	AAAAB	AAAAA	2.415	2.918	2.415	2.918	1.678	2.198
5	3	AAABC	AAABC	AAAAA	2.755	3.743	2.755	3.743	1.603	2.091
5	4	AAABC	AABBC	AAAAA	2.642	3.618	2.415	3.667	1.338	1.936
5	5	AABBC	AABBC	AAAAA	2.615	3.572	2.615	3.572	1.288	1.793
5	6	AABCD	AABBC	AAAAA	2.577	3.424	2.515	3.445	1.315	1.669
5	7	AABBC	AABCD	AAAAA	2.480	3.311	2.305	3.333	1.112	1.561
5	8	AABCD	AABCD	AAAAA	2.216	3.238	2.216	3.238	0.963	1.467
5	9	AABCD	ABCDE	AAAAA	2.219	3.144	2.052	3.168	0.850	1.385
5	10	AABCD	ABCDE	AAAAA	2.072	3.052	1.993	3.101	0.760	1.313
6	2	AAAAAB	AAAABB	AAAAAA	2.678	3.137	2.000	3.177	1.678	2.333
6	3	AAAAAB	AAAABC	AAAAAA	3.018	3.642	2.986	4.052	1.603	2.230
6	4	AAABBC	AAABBC	AAAAAA	2.897	3.979	2.897	3.979	1.490	2.079
6	5	AAAABC	AABBCC	AAAAAA	2.715	3.675	2.696	3.881	1.347	1.937
6	6	AAABBC	AABBCC	AAAAAA	2.770	3.714	2.708	3.779	1.315	1.812
6	7	AABBCC	AABBCC	AAAAAA	2.703	3.657	2.703	3.657	1.334	1.702
6	8	AABBCC	AABBCD	AAAAAA	2.620	3.530	2.548	3.549	1.156	1.606
6	9	AABBCC	AABBCD	AAAAAA	2.510	3.404	2.403	3.451	1.020	1.521
6	10	AABBCD	AABCDE	AAAAAA	2.345	3.354	2.259	3.375	0.912	1.446

Table 3: Initial guess structure, and the amount of information gained accordingly (in bits).

2.1 Rules

Rules for the new game are extremely simple. Unlike the "static" MasterMind, the game starts with an initial guess from the analyst. Thereafter, the setter must respond by choosing a hidden code and telling the analyst the number of correct and included colors from his guess, compared to the selected hidden code. At round i of the game, the analyst tries guess G_i^k to which the setter must respond by choosing a hidden code X_i and telling the analyst a response R_i^j. The code X_i, selected by the setter, must conform with all the previous responses to the analyst's guesses. In other words, the setter is not allowed to contradict his previous responses. The round goes on until the setter is compelled to surrender!! This occurs when he cannot find any code X_{m+1}, that is different from the analyst's final guess. In this case the game is said to be over after m steps.

	MaxMin Strategy	MaxEnt Strategy
- Average # of steps required to break the code (m)	3.860	3.835
- Standard deviation of m	0.548	0.623
- Average information gained in each guess in bits (I)	2.679	2.696
- Standard deviation of I	1.047	1.168
- Average information from the first guess (I1)	2.731	3.090
- Standard deviation of I1	0.905	1.088

Table 4: Results obtained from 500 sample games played under the MaxMin and MaxEnt strategies.

2.2 Formulation

Identically to the "static" case, the 3-tuples (P_{i-1}, G_i^k, R_i^j) may be viewed as a message $m_i^{k,j}$ from a zero memory information source, $\mathcal{S}$. The amount of information that the analyst gets from such messages, $I(m_i^{k,j})$, may be used to obtain a smaller pool $P_i \subset P_{i-1}$. The greater $I(m_i^{k,j})$ the smaller the size of the new pool will be. Therefore, in choosing guess G_i^k, the analyst must try to maximize $I(m_i^{k,j})$ while in choosing code X_i (and hence the response R_i^j) the setter must try to minimize $I(m_i^{k,j})$. The definition of $I(m_i^{k,j})$ is the same as that of the "static" case. As was the case in the "static" game and disregarding the selected guess and code, a positive amount of information is always gained. Therefore, the game will always end after a finite number of guesses, m, and it is the analyst's goal to minimize m, whereas it is the setter's goal to maximize m. The bounds ($m_{L.U.B.}$, and m_{max}) for m in the "Dynamic" game are the same as those of the "static" game.

The analogy we have drawn earlier between the MasterMind game and the Diagnosis process can be extended to the dynamic case as well. In this respect, the setter behaves like an *adversary* who is trying to hide from the diagnoser by responding to the tests in a non-contradictory fashion. Obviously, the goal of the diagnoser is to make the adversary surrender in the least possible number of rounds.

2.3 Setter and Analyst Strategies

It is interresting to note that it is the setter's response R_i^j, that determines the amount of information to be given to the analyst. Therefore, in each round i, the setter has to consider all possible responses and choose the one that offers to the analyst the minimum amount of information. Using this strategy the setter is guaranteed, in the worst case, of being able to prolong the game to its upper bound. Strategies of the analyst are the same as those presented in the "static" case. A complete description of the setter's strategy follows:-

```
Procedure Setter($P_{i-1}, G_i^k$)
  Begin
    - For all possible responses $R_i^j$
      - Compute I($m_i^{k,j}$)
    - Select response $R_i^o$ for which $I(m_i^{k,o}) \leq I(m_i^{k,x})$ ; where $x \neq o$.
  End.
```

2.4 Conclusion

The Dynamic MasterMind game, in which the setter becomes an active player, was introduced. Optimal strategies, for both the setter and the analyst, were developed on the basis of the number of elements in the solution pool one step ahead. Further developments for obtaining optimal strategies should consider not only the number but also the correlation between the members of the solution pool.

References

[1] Ash R. B., Information Theory , John Wiley & Sons, New York.

[2] Abramson N. M., Information Theory and Coding , McGraw-Hill Book Co., New York.