An experimental study of exhaustive solutions for the Mastermind puzzle

J.J. Merelo[1], A.M. Mora[1], C. Cotta[2], T. Runarsson[3,*]

1 Dept. of Computer Architecture and Technology, University of Granada, Granada, Spain
2 Dept. of Languages and Computer Sciences, University of Málaga, Málaga, Spain
3 School of Engineering and Natural Sciences, University of Iceland, Reykjavík, Iceland
∗ E-mail: jmerelo@geneura.ugr.es

Abstract

Mastermind is in essence a search problem in which a string of symbols that is kept secret must be found by sequentially playing strings that use the same alphabet, and using the responses that indicate how close are those other strings to the secret one as hints. Although it is commercialized as a game, it is a combinatorial problem of high complexity, with applications on fields that range from computer security to genomics. As such a kind of problem, there are no exact solutions; even exhaustive search methods rely on heuristics to choose, at every step, strings to obtain as a response the best possible hint; these methods mostly try to play the move that offers the best reduction in search space size in the next step; this move is chosen according to an empirical score. In this paper we will examine several state of the art exhaustive search methods and show that another factor, the presence of the actual solution among the candidate moves, or, in other words, the fact that the actual solution might obtain the highest score, plays also a very important role. Using that, we will propose new exhaustive search approaches that obtain results which are comparable to the classic ones, obtain results that are as good as the best of them, and, besides, are better suited as a basis for non-exhaustive search strategies such as evolutionary algorithms, since their behavior in a series of key indicators is better than the classical algorithms.

Introduction

Mastermind [1] is a board game that has enjoyed world-wide popularity in the last decades. Although its current version follows a design created in the 70s by the Israeli engineer Mordecai Meirowitz [2], the antecedents of the game can be traced back to traditional puzzles such as *bulls and cows* [3] or the so-called *AB* [4] game played in the Far East. Briefly, MasterMind is a two-player code-breaking game, or in some sense a single-player puzzle, where one of the players –the *codemaker* (CM)– has no other role in the game than setting a hidden combination, and automatically providing hints on how close the other player –the *codebreaker* (CB)– has come to correctly guess this combination. More precisely, the flow of the game is as follows:

- The CM (Alice in cryptography, [5]) sets and hides a length ℓ combination of κ symbols. Therefore, the CB is faced with κ^ℓ candidates for the hidden combination, which is typically represented by an array of pegs of different colors (but can also be represented using any digits or letter strings) and hidden from the CB.

- The CB (which is called Bob in cryptographic contexts such as [6]) tries to guess this secret code by producing a combination (which we will call move) with the same length, and using the same set of symbols. As a response to that move, the CM acting as an *oracle* (which explains the inclusion of this game in the category called *oracle* games) provides information on the number of symbols guessed in the right position (black pegs in the physical board game), and the number of symbols with the correct color, but in an incorrect position (white pegs); this is illustrated in Table 1.

Table 1. Progress in a MasterMind game that tries to guess the secret combination *ABBC*. 2nd and 4th combinations are not *consistent* with the first one, not coinciding in two positions and one color with it.

Combination	Response
AABB	2 black, 1 white
ABFE	2 black
ABBD	3 black
BBBE	2 black
ABBC	4 black

- The CB uses (or not, depending on the strategy he is following) this information to produce a new combination, that is assessed in the same way. If he correctly guesses the hidden combination in at most N attempts, the CB wins. Otherwise, the CM takes the game. N usually corresponds to the physical number of rows in the game board, which is equal to fifteen in the first commercial version.

- CM and CB are then interchanged, and several rounds of the game are played. This is a part of the game we do not consider. The player that is able to obtain the solution in the minimal amount of attempts wins.

In this paper, we will consider the puzzle only from the point of view of the CB , that is, a solver will be confronted with an oracle that will award, for each move, a number of black and white pegs. We will make no assumptions on the CM other that it will use random combinations (which in fact it does, although actual applications of Mastermind might use some information on the distribution of symbols [6]). A human player will probably have some kind of bias, but we are interested in finding the most general playing method without using that kind of information.

From the point of view mentioned in the last paragraph this puzzle is, in fact, a quite interesting combinatorial problem, as it relates to other *oracle*-type problems such as the hacking of the PIN codes used in bank ATMs [7, 8], uniquely identifying a person from queries to a genetic database [6, 9], or identifying which genotypes have interesting traits for selectively phenotyping them [10]; this is just a small sample of possible applications of the solution of the game of Mastermind. A more complete survey of applications (up to 2005) can be found in our previous paper [?], but recent posts in the questions-and-answers website StackOverflow and mentions in special issues devoted to *fun with algorithms* [11] are witness to the ongoing interest in finding solutions to the game [12].

Mastermind is also, as has been proved recently, a complex problem, paradigmatic of a whole class of search problems [13]; the problem of finding the solution has been shown to be NP-complete under different formulations [14, 15], and the problem of counting the number of solutions compatible with a certain set of answers is #P-complete [16]. This makes the search for good heuristic algorithms to solve it a challenge; besides, several issues remain open, such as what is the lowest average number of guesses needed to solve the problem for any given κ and ℓ. Associated to this, there arises the issue of coming up with an efficient mechanism for finding the hidden combination independently of the problem size, or at least, a method that scales gracefully when the problem size increases.

In this paper we will mainly examine empirical exhaustive search algorithms for sizes in which they are feasible and try to find out which are the main factors that contribute to finding the hidden combination in a particular (and small) number of moves; in this context, we mean by exhaustive algorithms those that examine, in turn, all items in the search space, discarding a few in each step until the solution (the secret code) is found. While most methods of this kind consider as the only important factor the reduction of the search space [17], we will prove that how different combinations, and in particular the secret (and unknown) one is scored contributes also to finding the solution faster. Taking into account how often

the hidden combination appears among the top scorers for a particular method, we will propose new exhaustive search methods that are competitive with the best solutions for Mastermind known so far. Besides, these methods do not add too much complexity to the solution and present certain behaviors that are better than the empirical solutions used so far.

It is obvious that exhaustive search methods can only be used for small sizes in NP-hard problems such as this one. However, its analysis and the proposal of new methods have traditionally been, and will also be used, to design metaheuristic search methods (such as evolutionary algorithms) that are able to search much further in the parameter space. Since that has been our line of research for a long time [?, ?, ?], eventually we will use them in those algorithms, but that is outside the scope of this paper.

The rest of the paper is organized as follows: we lay out the terminology and explain the solutions to the game of Mastermind in Section ; the state of the art in solutions to it is presented next, in Section . Next we will analyze the best performing Mastermind methods known so far in Section 1 and reach some conclusions on its way of working; the conclusions extracted in this analysis will be used in Section 1.2 to propose and analyze the new solutions proposed in this paper, which we have called Plus and Plus2. This Section will be followed by conclusions in the last Section 1.2.

The game of Mastermind

As mentioned in Section , a MasterMind problem instance is characterized by two parameters: the number of colors κ and the number of pegs ℓ. Let $\mathbb{N}_\kappa = \{1, 2, \cdots \kappa\}$ be the set of symbols used to denote the colors. Subsequently, any combination, either the hidden one or one played by the CB, is a string $c \in \mathbb{N}_\kappa^\ell$. Whenever the CB plays a combination c_p, a *response* $h(c_p, c_h) \in \mathbb{N}^2$ is obtained from the CM, where c_h is the hidden combination. A response $\langle b, w \rangle$ indicates that the c_p matches c_h in b positions, and there exist other w symbols in c_p present in c_h but in different positions.

Let us define *consistent* combinations [13] in the following way: a combination c is *consistent* with another played previously c_p if, and only if, $h(c, c_p) = h(c_p, c_h)$, i.e., if c has as many black and white pegs with respect to the c_p as c_p has with respect to the hidden combination. Intuitively, this captures the fact that c might be a potential candidate for secret code in light of the outcome of playing c_p. We can easily extend this notion and denote a combination c as consistent (or feasible) if, and only if, it is consistent with all the combinations played so far, i.e., $h(c, c_p^i) = h(c_p^i, c_h)$ for $1 \leqslant i \leqslant n$, where n is the number of combinations played so far, and c_p^i is the $i-$th combination played. Any consistent combination is a candidate solution. A *consistent set* is the set of all consistent combinations at a particular point in the game.

Consistent combinations are important because only a move using one of them decreases the size of the consistent set, at least by one (the combination itself). A non-consistent combination might or might not do that, depending on its similarity to already played moves. So most Mastermind strategies play a consistent combination. We will see next how these concepts are used to find the secret combination in a minimum number of combinations.

State of the art

Most theoretical work done on the game of Mastermind tries to find upper bounds to the number of moves needed to find the solution. Best bounds known so far [18] point at $O(\ell log \ell)$, which is slightly faster than linear, and even $O(\ell log log \ell)$ when $\ell = \kappa$. However, the proof of this bound, although useful and the tighter so far, does not provide a valid approach to choosing the next string to play, since it is not based on the use of the consistent set, but on the deterministic reduction of search space via the elimination of colors.

Most methods for solving Mastermind, however, use the set (or a subset) of consistent solutions and different heuristics to choose which combination is played. A simple strategy, valid at all levels, eschews looking for sets of consistent combinations and plays the first consistent combination that is found (be it in an enumerative search, or after random draws from the search space, or searching for it using evolutionary algorithms [?]). In general, we will call this strategy Random; in fact, the behavior of all these strategies is statistically indistinguishable; that is, it does not matter the method you choose for finding a consistent string. This strategy, though valid, does not offer the best results. However, it depends on randon draws from the set of valid strings and thus can be extended to arbitrary sizes. If a method to keep the used strings is added, it is guaranteed to find the solution (using an amount of memory that grows with search space size).

Table 2. Table of partitions after two combinations have been played; this table is the result of comparing each combination against all the rest of the set, which is the set of consistent combinations in a game after two combinations have already been played. In boldface, the combinations which have the minimal worst set size (which happen to be in the 1b-1w column, but it could be any one); in this case, equal to ten. A strategy that tries to minimize worst case would play one of those combinations. The column 0b-1w with all values equal to 0 has been suppressed; column for combination 3b-1w, being impossible, is not shown either. Some rows have also been eliminated for the sake of compactness.

Combination	Number of combinations in the partition with response									
	0b-2w	0b-3w	0b-4w	1b-1w	1b-2w	1b-3w	2b-0w	2b-1w	2b-2w	3b-0w
AABA	0	0	0	14	8	0	13	1	0	3
AACC	8	0	0	10	5	0	8	4	1	3
AACD	6	2	0	11	6	1	4	5	1	3
AACE	6	2	0	11	6	1	4	5	1	3
AACF	6	2	0	11	6	1	4	5	1	3
ABAB	8	0	0	10	5	0	8	4	1	3
ABAD	6	2	0	11	6	1	4	5	1	3
ABAE	6	2	0	11	6	1	4	5	1	3
ABAF	6	2	0	11	6	1	4	5	1	3
ABBC	4	4	0	10	8	0	8	1	1	3
ABDC	3	4	1	11	9	1	4	2	1	3
ABEC	3	4	1	11	9	1	4	2	1	3
ABFC	3	4	1	11	9	1	4	2	1	3
ACAA	0	0	0	14	8	0	13	1	0	3
ACCB	4	4	0	10	8	0	8	1	1	3
ACDA	0	0	0	16	10	2	5	3	0	3
BBAA	8	0	0	10	5	0	8	4	1	3
BCCA	4	4	0	10	8	0	8	1	1	3
BDCA	3	4	1	11	9	1	4	2	1	3
BECA	3	4	1	11	9	1	4	2	1	3
BFCA	3	4	1	11	9	1	4	2	1	3
CACA	8	0	0	10	5	0	8	4	1	3
CBBA	4	4	0	10	8	0	8	1	1	3
CBDA	3	4	1	11	9	1	4	2	1	3
CBEA	3	4	1	11	9	1	4	2	1	3
CBFA	3	4	1	11	9	1	4	2	1	3
DACA	6	2	0	11	6	1	4	5	1	3
EBAA	6	2	0	11	6	1	4	5	1	3
FACA	6	2	0	11	6	1	4	5	1	3
FBAA	6	2	0	11	6	1	4	5	1	3

And it does not do so because not all combinations in the consistent are able to reduce its size in the next move in the same way. So other not so naÃ⁻ve solutions concentrate on scoring all combinations in the consistent set according to a heuristic method, and playing one of the combinations that reaches the top score, a random one or the first one in lexicographical order. This score is always based on the concept of Hash Collision Groups, HCG [4] or *partitions* [17]. All combinations in the consistent set are compared with each other, considering one the secret code and the other a candidate solution; all combinations will be grouped in sets according to how they compare to a particular one, as shown in Table 2. For instance, in such table there is a big set of combinations whose response is exactly the same: ABDC, ABED, ABFC, ABAD, ABAE, ABAF... All these combinations constitute a *partition*, and their score will be exactly the same.

To formalize these ideas, let $\vec{\Xi} = \{\Xi_{ibw}\}$ be a three-dimensional matrix that estimates the number Ξ_{ibw} of combinations that will remain feasible after combination c_i is played and response $\langle b, w \rangle$ is obtained from the CM. Then, the potential strategies for the CB are:

1. Minimizing the worst-case partition [19]: pick $c_i = \arg\min_i\{\max_{b,w}(\Xi_{ibw})\}$. For instance, in the

set in Table 2 this algorithm would play one of the combinations shown in boldface (the first one in lexicographical order, since it is a deterministic algorithm).

2. Minimizing the average-case partition [20, 21]: pick $c_i = \arg\min_i\{\sum_{b,w} p_{bw}\Xi_{ibw}\}$, where p_{bw} is the prior probability of obtaining a particular outcome. If for instance we compute $p_{bw} = \sum_i \Xi_{ibw} / \sum_{i,b,w} \Xi_{ibw}$, then AACD would be the combination played among those in Table 2.

3. Maximizing the number of potential partitions [17]: pick $c_i = \arg\max_i\{|\{\Xi_{ibw} > 0\}|\}$, where $|C|$ is the cardinality of set C. For example, a combination such as ABDC in Table 2 would result in no empty partition. This strategy is also called *Most Parts*.

4. Maximizing the information gained [?, 22, 23]: pick $c_i = \arg\max_i\{H_{b,w}(\Xi_{ibw})\}$, where $H_{b,w}(\Xi_{i[\cdot][\cdot]})$ is the entropy of the corresponding sub-matrix. We will call this strategy Entropy.

Algorithm 1: Choosing the next move in the general case

---

1 typedef Combination: **vector**$[1..\ell]$ **of** $\mathbb{N}_\kappa$;

2 procedure NextMove (**in:** F: List[Combination], **out:** *guess*: Combination);

3 var TopScorers: List[Combination];

4 Score(F);

5 *guess* $\leftarrow$ RandomElement (TopScorers);

All strategies based on partitions work as follows:

1. Score all combinations in the consistent set according to the method chosen (Entropy, Most Parts, Best Expected, or Minimize Worst)

2. Play one of the combinations with the best score

3. Get the response from the codemaker, and unless it is *all blacks*, go to the first step.

This is also represented more formally in Algorithm 1, where Score scores all combinations according to the criterion chosen, and RandomElement extracts a random element from the list passed as an argument using uniform distribution.

Exhaustive strategies have only (as far as we know) been examined and compared for the base case of $\kappa = 6, \ell = 4$, for instance in [?]; restricted versions of the game have been examined for other spaces, for instance in [13]. The best results are obtained by Entropy and Most Parts, but its difference is not statistically significant. All other results (including the classical one proposed by Knuth [19]) are statistically worse. That is why in this paper we will concentrate on these two strategies, which represent the state of the art in exhaustive search. Bear in mind that all these strategies are empirical, in the sense that they are based in an assumption of how the reduction of the search space work; there is, for the time being, no other way of proposing new strategies for the game of mastermind.

Besides, the size of the the search space is used as proxy for what is actually the key to success of a strategy: the probability of drawing the winning combination at each step. It is evident that reducing the number of combinations will eventually result in drawing the secret one with probability one. However, it should not be neglected that the probability of drawing it even if it is not the only remaining combination is non-zero and care should be taken so that this probability is either maximized, or at least considered to minimize the number of moves needed to find it. So far, and to the best of our knowledge, no study has been made of this probability; we will show in this paper its influence on the success of a strategy.

Let us finally draw our attention to the first move. It is obviously an important part of the game, and since, a priori, all combinations are consistent a strategy would consist in using all combinations in the

game and playing one according to its score. However, this is not a sensible strategy even for the smaller size; *consistency* does not make any sense in absence of responses, since the partitions will not hold any information on the secret code. So, in most cases, a fixed move is used, and the one proposed by Knuth (according to its own Minimize Worst strategy) is most usually employed. Since Knuth's strategy [19] was created for $\kappa = 6, \ell = 4$ it is difficult to extrapolate it to higher dimensions, so papers vary in which combination is used. At any rate, the influence of the first move will be felt mainly in the reduction of size achieved *before* the first empirical move is made (second move of the game), but a good reduction will imply a significant change in the average number of games. We will bear this in mind when testing the different empirical strategies.

Essentially, then, Most Parts and Entropy are the state of the art in exhaustive search strategies. These algorithms, along with all others used in this paper, are written in Perl, released under a open source licence and can be downloaded from `http://goo.gl/G9mzZ`. All parameters, experiment scripts, data extraction scripts and R data files can be found within the `app/IEEE-CIG` directory. Next we will examine how these algorithms work in two different sizes, which are considered enough to assess its performance; in Section 1.2 we will propose new methods and prove that they obtain a better average number of moves making them the new state of the art in exhaustive search algorithms for the game of Mastermind.

1 Analyzing exhaustive search methods

In this section, we will outline a methodology for analyzing exhaustive solutions to the game of Mastermind and apply it to two strategies that are usually considered the best, Entropy and Most Parts, for two different problem sizes, keeping $\ell = 4$ fixed and setting $\kappa = 6$ (Subsection 1.1) and $\kappa = 8$ (Subsection 1.2).

1.1 $\ell = 4, \kappa = 6$

Let us look first at the smallest usual size, $\kappa = 6$ and $\ell = 4$. The best two strategies in this case have been proved to be Entropy and Most Parts [?, 20], in both cases starting using Knuth's rule, ABCA [19]. An analysis of the number of moves obtained by each method are shown in Table 3. To compute this

Table 3. Average (with error of the mean) and Maximum number of moves for two search strategies: Most Parts and Entropy.

Method	Number of moves		
	Average	Maximum	Median
Entropy	4.413 ± 0.006	6	4
Most Parts	4.406 ± 0.007	7	4

average, 10 runs over the whole combination space were made. In fact the difference in the number of moves is small enough to not be statistically significant (using Wilcoxon test, p-value = 0.4577), but there is a difference among them, the most striking being that, even if the maximum number of moves is higher for Most Parts, Entropy has a higher average.

To check where that difference lies we will plot a histogram of the number of moves needed to find the solution for both methods, see Fig. 1.

As should be expected for the negligible difference in the average number of moves, differences here are very small. It is noticeable, however, than Most Parts finishes less times in 1 to 3 moves, and also 5 moves. Most parts only finishes in less occasions than Entropy for 4 moves; quite obviously, too, there are a few times in which Most Parts needs 7 moves, but only 8 out of the total 12960 games (10 games
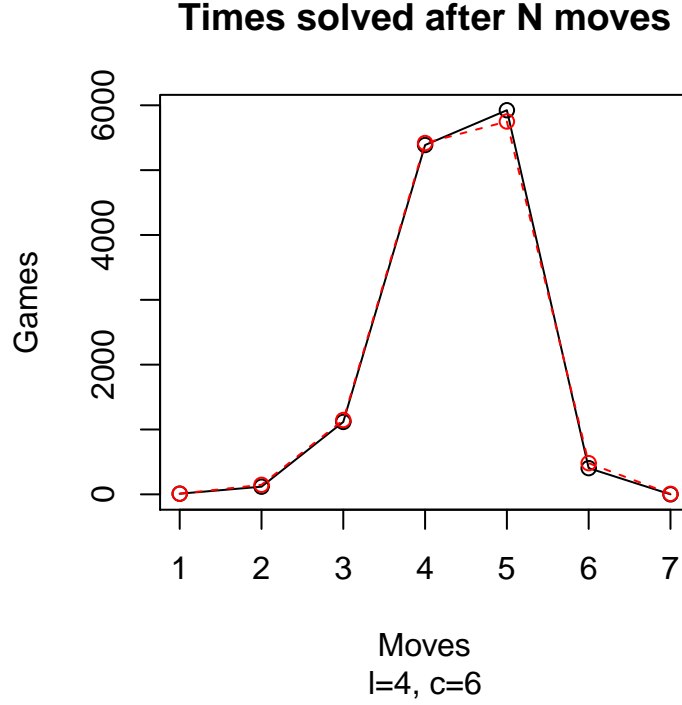
Times solved after N moves

Figure 1. Histogram with the count of the number of times every method is able to find the solution in every number of moves, for the Entropy method (black and solid) and Most Parts (light or red, dashed).

times the total number of combinations, 1296). Again, differences are not significant, but noticeable , so we will try to seek what is its source; these differences must be the cause why eventually Most Parts achieves a slightly higher average number of moves. Let us look further into these differences by plotting the difference in the number of games won by every one in a particular number of moves, let's say x. That is shown in Figure 2. For all moves, except 5 and 7 (by that move Entropy has always finished), Most Parts has more games. Since both methods play the same number of games, the difference is offset by the number of games more, around 150, that Entropy finishes by move number five; this is also seen clearly in the histogram 1. However, the important issue is the difference in *score*, that is, in number of moves brought by finishing each move. Differences in scores are computed by multiplying the difference in the number of games by the move number, and plotted in Figure 3. It obviously follows the same pattern than Figure 2, however it gives us an idea of the contribution of differences to total score. Since in this case the total score for Entropy is 57189 and for Most Parts 57106, and Most Parts is always better (negative difference in score) than Entropy, we see that for low number of moves Most Parts is accumulating differences, although it is worsened by not being able to finish as many times as Entropy in 5 moves. This is the key move, and the shape of the plot indicates a change of regime in the game. It also shows that no single strategy is better than the other, yielding the non-significant difference in moves (and scant difference in score, less than 100 over 12960 games!). The main conclusion would be that , in fact, bot solutions are very similar, and this is also supported by other experiments (not shown here) with more games, that do not yield a significant difference either. However, we can also see that the way they find the solution is different, that is why we will study other aspects of the algorithm in the
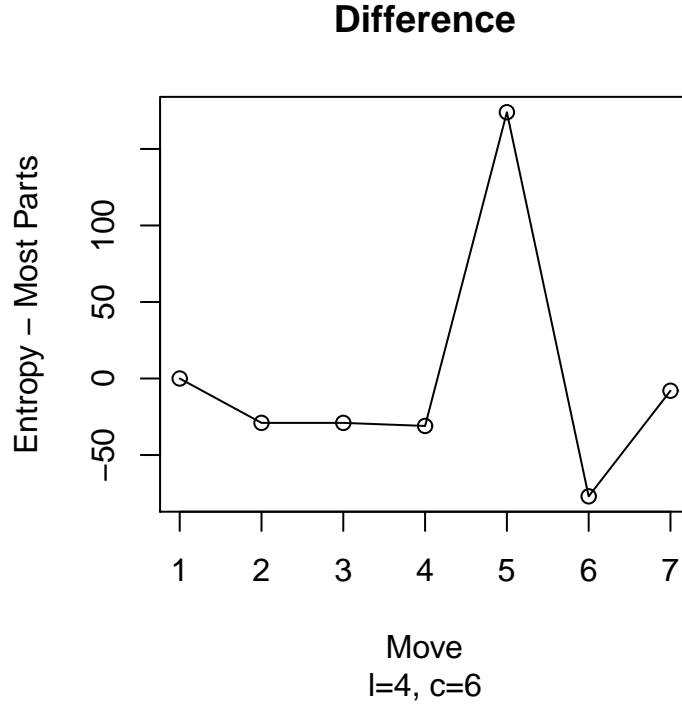
Difference

Figure 2. .

next paragraphs.

Table 4. Average and standard deviation of the number of combinations remaining after every move (or step). The numbers are the same after the first move (not shown here), since they are playing the same one.

Before move #	Entropy	Most Parts
3	23 ± 14	24 ± 15
4	3.1 ± 1.8	3.4 ± 2.4
5	1.13 ± 0.35	1.17 ± 0.43
6	1	1.02 ± 0.17
7		1

Since both methods try to reduce the size of the consistent set, we will look at their size and how it changes with the number of moves. We will log the size of the remaining consistent combinations at every step, and this is shown in Fig. 4 and Table 4.

In effect, the Entropy method is more efficient in reducing the size of the set of remaining solutions, which is the reason why usually it is presented as the best method for solving Mastermind, since it achieves most effectively what it is intended to achieve: reduction in search space size. At every step, the difference is significant (using Wilcox test). This difference is of a single combination at the beginning, and decreases with time while still being significant; however, this reduction at the first stages of the game makes the search simpler and would, in principle, imply an easy victory for the Entropy technique,
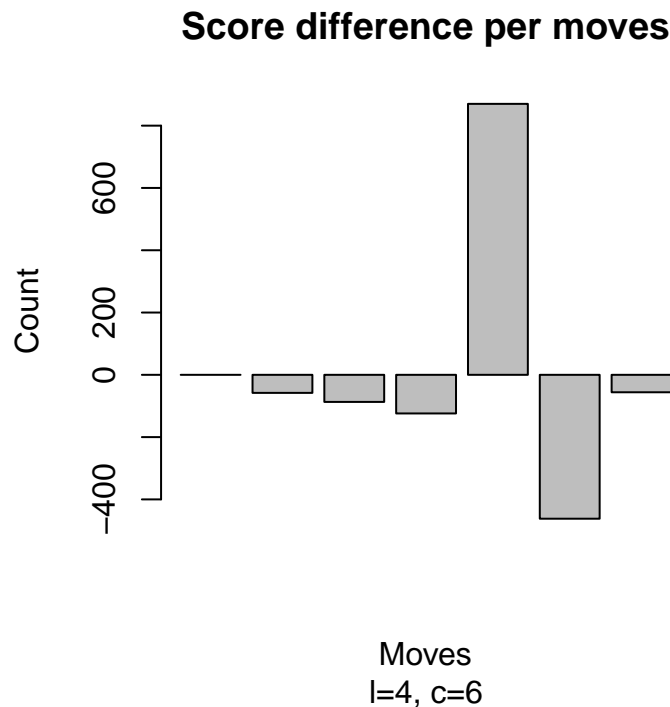
Score difference per moves

Figure 3. Difference in score among the number of games won by Entropy and Most Parts in x moves. Every y value in the graph 2 has been multiplied by the number of moves, resulting in the actual score the player would achieve. Remember that less is better; positive values mean that Entropy beats Most Parts (for that move), and vice versa.

as would be expected. However, the result is a statistical draw, with a slight advantage for the other method, Most Parts. Besides, this reduction does not explain the advantage found in Figures 1 and 2: if the number of solutions that remain is bigger (on average), why is Most Parts able to find the solution at those same stages more often than Entropy, which seems to be the key to success?

To discover why this happen, we will look at another result. As explained above, all methods are based on scoring consistent combinations according to the partitioning of the set they yield, and then playing randomly one of the combinations with the top score. If, by chance, the winning combination is in that set of top scorers, there is a non-zero possibility of playing it as next move and thus winning the match. In effect, what we want is to maximize at every step the probability of drawing the secret code. We will then look at whether the winning combination effectively is or not among the top scorers at each step. Results are shown in Fig. 5 and Table 5.

Table 5 clearly shows that the probability of finding the hidden combination among the top scorers increases with time, so that in the 5th move is practically one. But it also shows that (in the first moves) there is almost double the chance of having the winning combination among the top ones for Most Parts than for Entropy, so, effectively, and obviously depending on the size of the consistent set found at Table 4, the likelihood of playing the winning combination is higher for Most Parts and its key to success. Making a back of the envelope calculation, when the game arrives at the second move, which roughly 11000 games do, a third of them will include the winning combination among the top scorers, which again is roughly three thousand.
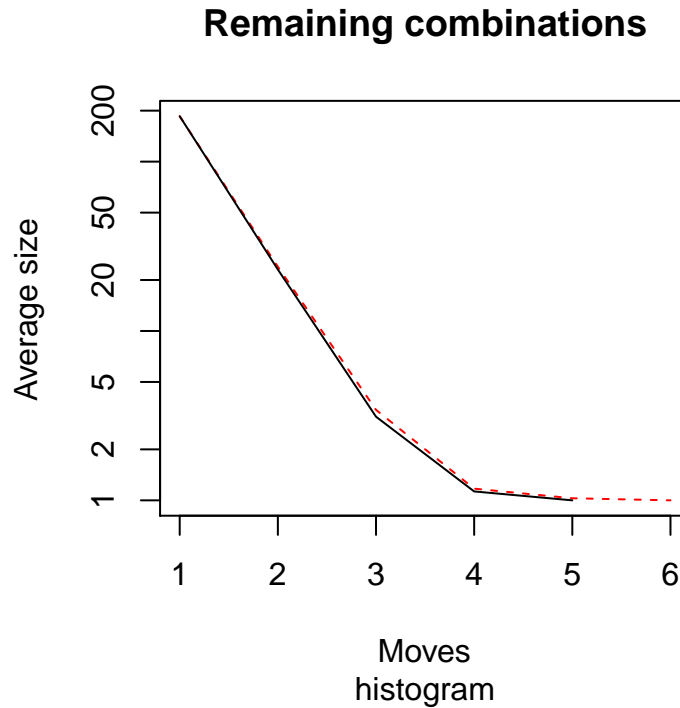
Remaining combinations

Figure 4. Average number of combinations (or moves) remaining after each move, for the Entropy method (black and solid) and Most Parts (light or red). This plot corresponds to the numbers represented in Table 4. The y axis is logarithmic for clarity, and the x axis is shifted by one (1 means the second move, size before the first move is always the whole space).

Table 5. Percentage of times the secret code is among the top scorers for each method.

In move #	Entropy	Most Parts
2	0.1142857	0.3644788
3	0.2919495	0.5270987
4	0.7721438	0.8242563
5	0.9834515	0.9810066
6		1

This is not the whole picture, however. Every set of top scorers will have a different size; even if the probability of finding the secret score among that set is higher, the probability of *drawing* it will be different if the size is smaller. The average of this probability has been drawn in Figure 6. While initially the probability is very small (but different), Most Parts has a better chance of obtaining the secret code at each move than Entropy, at least until move number 6 (label 4). That change of regime is reflected in the previous plots by a sudden increase in the number of games Entropy finishes (see Figures 1 and 2). However, since by that time Most Parts has been able to finish a good amount of times, the difference is not too big. Looking at figure 4 and table 4, which plots the average size of the set of remaining combinations, we see that by the 5th move (label 4) there is, almost always, a single remaining combination, but it happens more often for Entropy than for Most Parts; that is, in this move is when
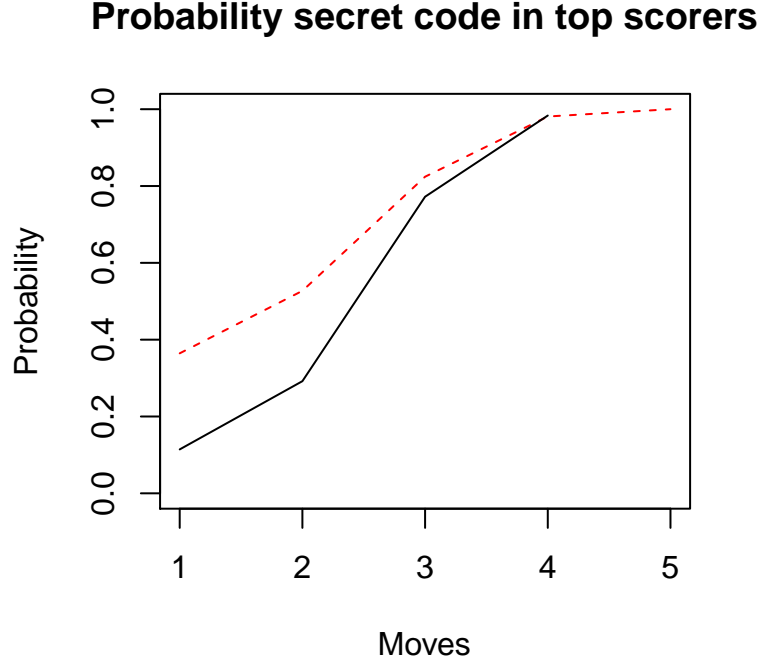
Probability secret code in top scorers

Figure 5. Chance of finding the secret code among the top scorers for the Entropy method (black and solid) and Most Parts (light or red). This plot corresponds to the numbers represented in Table 5.

the reduction of the search space effectively kicks in, accounting for the *change in regime* we mentioned before.

This, in turn, implies one of the main results of this paper: there are two factors in the success of a method for playing Mastermind. The first is the reduction it achieves on the search space size by playing combinations that reduce it maximally, but there is a second and non-negligible factor: the chance of playing the winning combination by having it among the top scorers. It can be said that there is a particular number of moves, which in this case is after the median, 4 moves, where a change of regime takes place. Before and up to that number of games, if a method finishes it is mainly due to drawing, by chance, the secret code. After the median the secret code is found because it is the only remaining element in the search space.

Most studies so far, however, had only looked at the smallest size. Let us study another problem size, with search space 4 times as big in the next subsection.

1.2 $\ell = 4, \kappa = 8$

The way the solutions work will change when the problem size is increased, so we will perform the same measurements for problem size $\kappa = 8, \ell = 4$. Search space size is four times as big; time to solution grows faster than lineally so it is not practical (although possible) to work with exhaustive search for sizes bigger than that; in fact, $\ell = 5, \kappa = 6$ is the next step that is feasible, but initial work has shown that the behavior is not different from this case, and it takes around two times as much; while the exhaustive solution to the smallest Mastermind size considered takes around half a second, it takes around 5 seconds
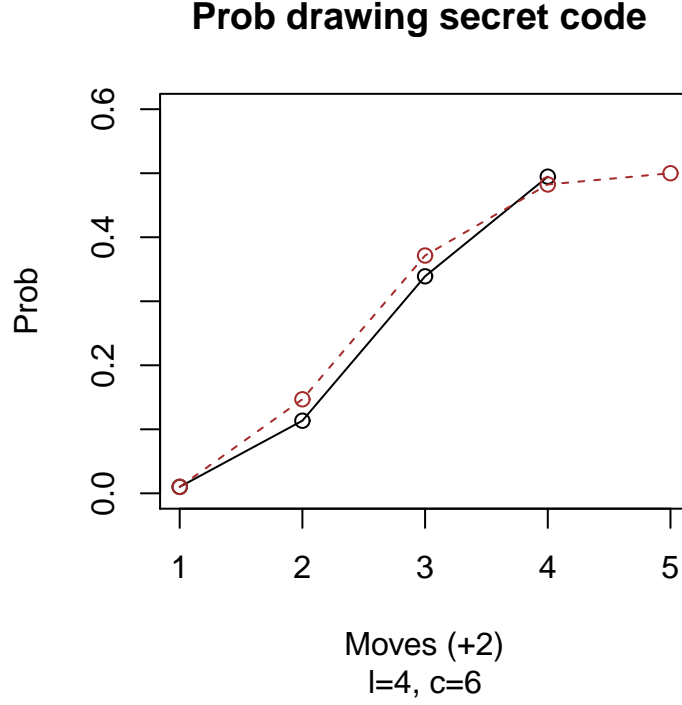
Prob drawing secret code

Figure 6. Chance of drawing the secret code for the Entropy method (black and solid) and Most Parts (light or red).

for $\kappa = 8, \ell = 4$, and 10 seconds for $\kappa = 6, \ell = 5$. In practice, this means that instead of using the whole search space 10 times over to compute this average, we will generate a particular set of 5000 combinations, which includes all combinations at least once and none more than two times. This instance set is available at the method website (`http://goo.gl/ONYLF`). A solution is searched, then, for every one of these combinations. We are playing ABCD as first move, as one interpretation of Knuth's [19] first move would say; that is, we play half the alphabet and start again by the first letter (ABCA would play half the ABCDEF alphabet and then start again).

The average number of moves is represented in Table 6. This confirms in parts our above hypothesis: the better capability Entropy has to decrease the size of the search space gives it an advantage in the average number of moves, keeping at the same time the ability of solving it in less maximum number of moves. However, for this number of experiments, the difference is significant (Wilcoxon paired test = 0.052). We will check whether this difference has the same origin as we hypothesized for the smaller search space size, by looking at the same variables.

The equivalent to Fig. 1 has been represented in Fig 7. The shape is similar, but the solid line that represents Entropy is slightly below the dashed line for Most Parts for the highest number of moves which accounts for the small difference in average number of moves. Is this difference accounted for by the size of the search space after each move? As previously, we will plot it in Fig. 8 and Table 8. Differences for all moves are significant according to Wilcoxon test, which means that actually Entropy is achieving what it was designed for: reduce in a significant way the search space, until the secret combination is found. However, in the same way as has been shown above, other mechanisms are also at work to find the solution before that reduction.

Table 6. Average (with error of the mean) and Maximum number of moves for two search strategies: Most Parts and Entropy for $\kappa = 8, \ell = 4$.

Method	Number of moves		
	Average	*Maximum*	*Median*
Entropy	5.132 ± 0.012	8	5
Most Parts	5.167 ± 0.012	8	5

Times solved after N moves
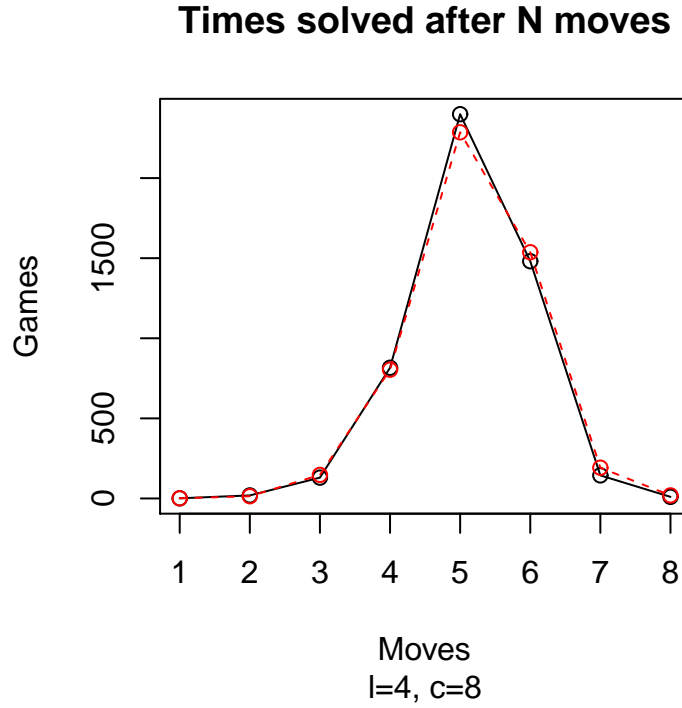
Figure 7. Frequency of the number of moves up to (and including) the secret combination, for $\kappa = 8, \ell = 4$. As usual, red or light dashed line represents Most Parts and solid black line Entropy.

Table 7. Average and standard deviation of the number of combinations remaining after every move. The number of combinations is the same after the first move (not shown here), since both are playing the same first move.

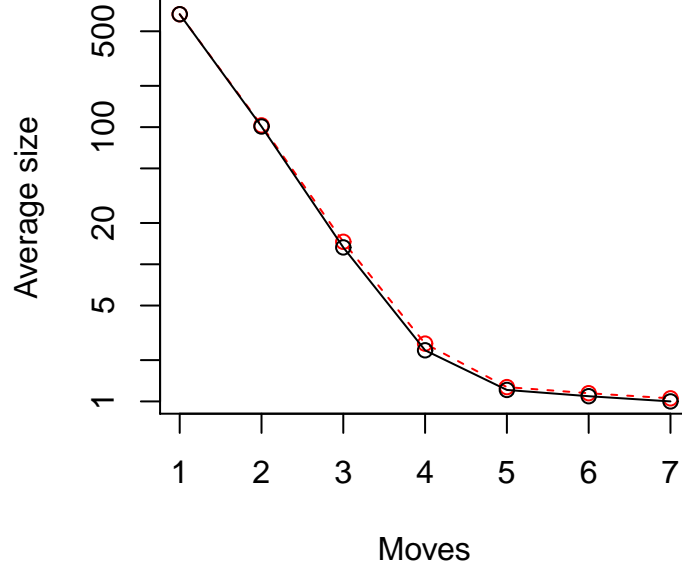| After move #	Entropy (ABCD)	Entropy (ABCA)
2	98 ± 67	102 ± 69
3	13 ± 10	14 ± 12
4	2.4 ± 1.6	2.63 ± 1.96
5	1.21 ± 0.46	1.27 ± 0.52
6	1.07 ± 0.27	1.08 ± 0.31
7	1	1.25 ± 0.46

Remaining combinations, average size

Figure 8. Average size of the consistent set, that is, the set of solutions that have not been discarded at a point in the game for $\kappa = 8, \ell = 4$. As usual, red or light dashed line represents Most Parts and solid black Entropy.

However, we will have to look in this case too at the differences in games and score by both methods with is represented in Figures 9 and 10. The first one represents the raw difference in number of times every one wins; again there is a change in behavior or phase shift (which we have called before a change of regime) when arriving at the fifth move, which is also around the median (exactly the median, in this case; it was one less than the median in the previous study). The scenario drawn in 10 is also similar, but shows that Entropy is able to beat Most Parts mainly because it is able to finish more times in less moves (around 5) than Most Parts (which accumulates lots of bad games that need 6,7 and 8 moves). The final difference is around 200 points for the 5000 games (25834 vs. 25662) which is very small but, in this case, significant.

We mentioned in Subsection 1.1 that the probability of finding the secret code among the top scorers was one of those features. It should be expected that the probability of finding the secret code among the top scorers will change; since the number of elements in the consistent set intuitively we should expect it to decrease. But this intuition is wrong, as shown in Table 8 and Fig. 11. In fact, if we compare these results with those shown in Table 5 we see that, for the same move, the proportion of times in which the secret code is among the top scorers is almost twice as big at the beginning for Most Parts and almost three times as big for Entropy. Interestingly enough, this also implies that, while for $\kappa = 6, \ell = 4$ this probability was three times as high for Most Parts, it is only two times as high now.

This decrease in the chance of finding the secret code might explain the difference in the average number of moves needed to find the solution, which for this size tilts the balance in the direction of Entropy. While for the smaller size this probability was enough to compensate the superior capability of
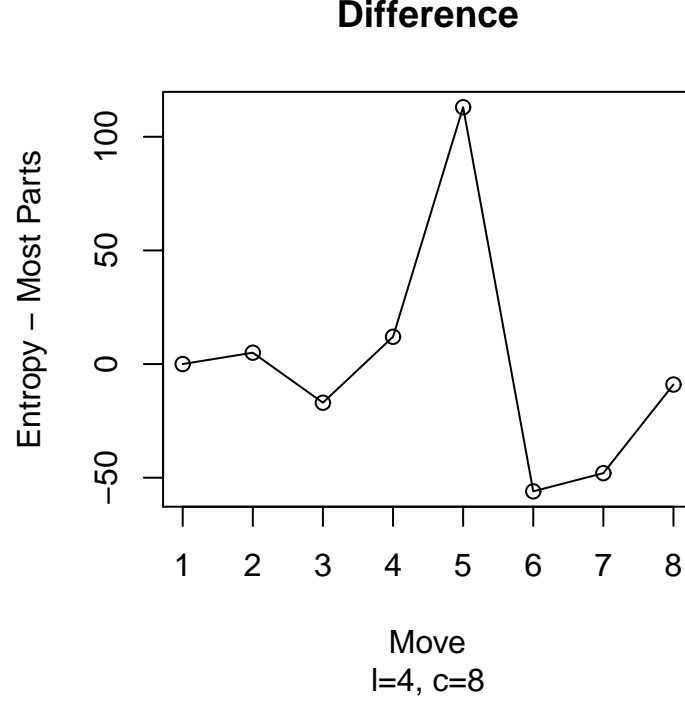
Figure 9. Difference among the number of games won by Entropy and Most Parts in x moves for $\kappa = 8, \ell = 4$.

the Entropy method in reducing the size of the search space, in this case the difference is not so high, which makes the Entropy method find the solution, on average, on less moves. This is actually only one of the factors in the actual probability of drawing the secret code, which is shown in Figure 12. In the same fashion as the previous study (shown in Figure 6), Most Parts is better in playing the secret combination up to the sixth move (label 5 in the graphic). But, after that, probabilities are similar and the change in regime takes place; in fact, looking at 9 and 8, it could be argued that the fact that after the 6th move almost always there is only one combination left accounts for the observed phase shift.

As we concluded the previous section, it is non-negligible the influence of the probability of a score that is able to consistently give top score to the secret combination; however, in this case the size of the search space implies that the probability of playing the secret code is smaller (by the 3rd move in $\kappa = 6$ it is as high as the 4th move in $\kappa = 8$), and thus its influence is not enough to make Most Parts as good

Table 8. Percentage of times the secret code is among the top scorers for each method, $\kappa = 8, \ell = 4$.

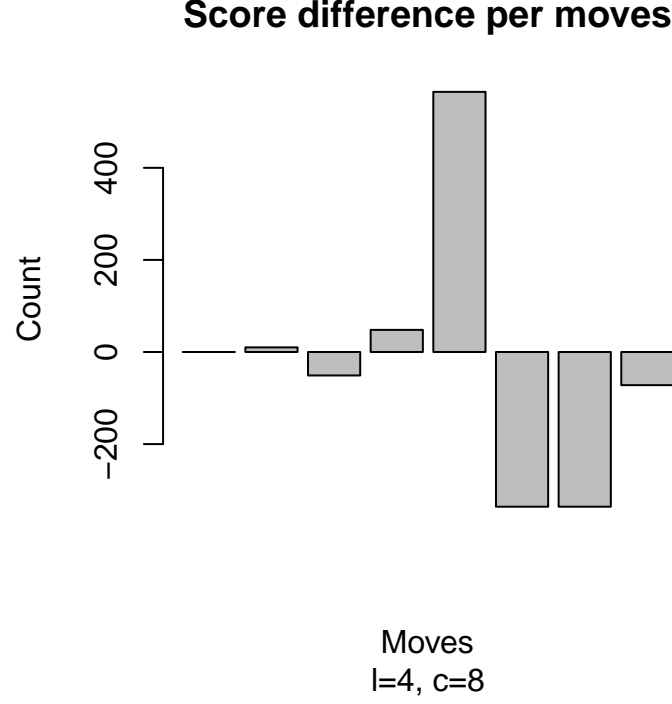| In move #	Entropy	Most Parts
2	0.3008602	0.6379276
3	0.3425758	0.6957395
4	0.4937768	0.7845089
5	0.8630084	0.9160276
6	0.9898990	1

Score difference per moves

Figure 10. Difference in score among the number of games won by Entropy and Most Parts in x moves for $\kappa = 8, \ell = 4$. Method is the same as for figure 3, that is, product of difference and number of moves.

as Entropy.

At any rate, it is clear that, also for this size, the only determinant factor is not the decrease in size brought by the method, but also the ability to score correctly. This, in turn, can take us to design new heuristic methods that are able to address both issues at the same time, and also study the difference brought by a different initial move.

New exhaustive search methods

Our intention by proposing new methods is to use the two main factors that we have proved have an influence in the performance of Mastermind-playing methods to devise new strategies that could serve as another empirical proof of the mechanism, and also, if possible, obtain methods that are able to obtain better results (or, at least, not worse results).

A new method for Mastermind will somehow have to match the two capabilities featured by Entropy and Most Parts: the first reduces competently the search space, and the second is able to score better the actual secret combination, which can then be selected for playing giving winning moves. So we will propose several methods that will try to combine them with the objective of achieving a better average number of moves.

The first one, which we will call *Plus*, works as follows (please see Algorithm 2): the set of consistent combinations is scored according to *both* Most Parts and Entropy. The sets of combinations with the top score according to both methods are extracted. If its intersection is non-null, a random combination

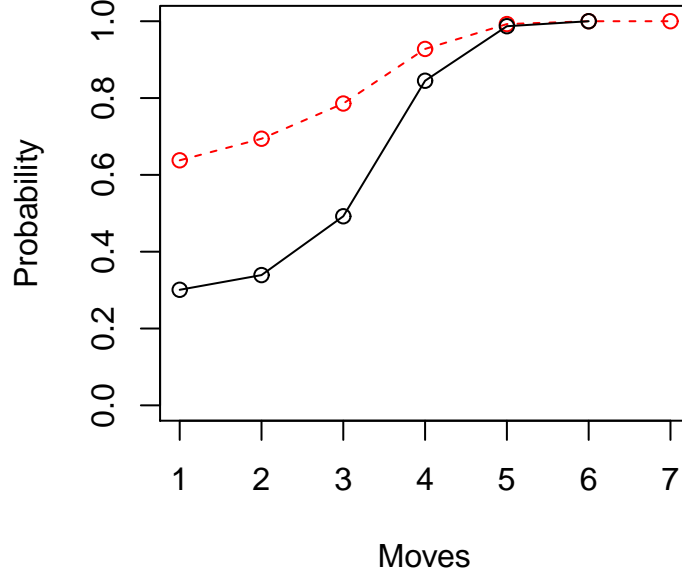**Prob. secret code in top scorers l=4 c=8**

Figure 11. Chance of finding the secret code among the top scorers for the Entropy method (black and solid) and Most Parts (light or red). This plot corresponds to the numbers represented in table 8 for $\kappa = 8, \ell = 4$.

Algorithm 2: Choosing the next move in the *Plus* Mastermind solution method. This algorithm is the new version we propose to the NEXTMOVE function presented in Algorithm 1.

---

1 **typedef** Combination: **vector**$[1..\ell]$ **of** $\mathbb{N}_\kappa$;

2 **procedure** NEXTPLUS (**in:** F: List[Combination], **out:** *guess*: Combination);

3 **var** TopScorersEntropy, TopScorersMostParts, TopScorersAll: List[Combination];

4 ENTROPYSCORE(F);

5 MOSTPARTSSCORE(F);

6 TopScorersEntropy$\leftarrow$ TOPSCORERSENTROPY(F);

7 TopScorersMostParts$\leftarrow$ TOPSCORERSMOSTPARTS(F);

8 TopScorersAll $\leftarrow$ TopScorersEntropy $\cap$ TopScorersMostParts;

9 *guess* $\leftarrow$ RANDOMELEMENT (TopScorersAll);

---

from it is returned. If it is null, a random string of the union of both sets is returned.

What we want to achieve with this method is a reduction of the consistent set in the same way as Entropy, but, by intersecting it with the set of top scorers for Most Part, the probability of finding the winning combination among them is also increased.

There is no single way of combining the two scoring methods. A second one tested, which we will call *Plus2* (and which is outlined in 3), works similarly. It proceeds initially as the Entropy method, by
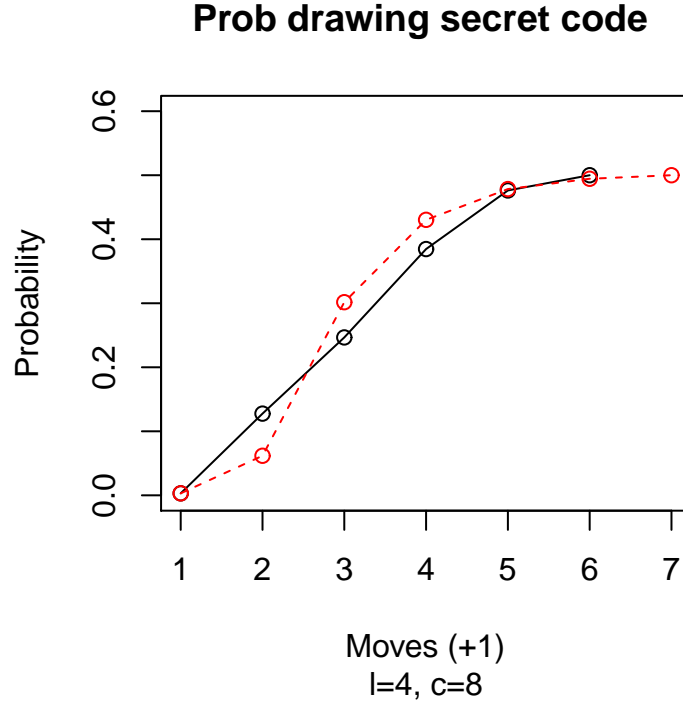
Prob drawing secret code

Figure 12. Chance of drawing the secret code for the Entropy method (black and solid) and Most Parts (light or red) for $\kappa = 8, \ell = 4$. Remember that the x axis is shifted by one, since the first move is fixed.

Algorithm 3: Choosing the next move in the *Plus* Mastermind solution method. Please note that up to line 6, this algorithm is identical to 2.

---

1 **typedef** Combination: **vector**$[1..\ell]$ **of** $\mathbb{N}_\kappa$;

2 **procedure** NEXTPLUS2 (**in:** F: List[Combination], **out:** *guess*: Combination);

3 **var** TopScorersEntropy, TopScorersMostParts, TopScorersAll: List[Combination];

4 ENTROPYSCORE(F);

5 MOSTPARTSSCORE(F);

6 TopScorersEntropy← TOPSCORERSENTROPY(F);

7 TopScorersAll← TOPSCORERSMOSTPARTS(TopScorersEntropy);

8 *guess* ← RANDOMELEMENT (TopScorersAll);

---

scoring combinations according to its partition entropy. But then, the top scorers of this method are again scored according to Most Parts. Out of the subset of top Entropy scorers with the highest Most Parts score, a random combination is returned. Please note that, in order to compute the Most Parts score of the top Entropy scorers, the whole Consistent Set F must be scored. In 3, line 7 would extract the top scorers according to the Most Parts method from the set of top scorers by the Entropy method. While in Plus what is played belongs to a subset of Entropy and Most Parts only if their intersection is non-zero, in Plus2 the new set of top scorers is *always* a subset of the top scorers for Entropy, further

scored using Most Parts.

As intended in Plus, Plus2 tries to reduce even more the consistent set size by narrowing down the set of combinations to those that have only top scores using both methods. In principle, the *priority* of Entropy and Most Parts can be swapped, by first scoring according to Most Parts and then choosing those with the best Entropy score. This method was also tested, but initial results were not as good, so we will show only results for these two and compare them with the traditional methods analyzed in the previous Section 1

Let us first look at the most important result: the number of moves. They are shown in Table 9. Difference is not statistically significant, but as indicated in the study presented in the previous section,

Table 9. Average and maximum number of moves for the new search strategies proposed, and comparison with the previous ones.

Method	Number of moves: Average	Number of moves: Maximum
Entropy	4.413 ± 0.006	6
Most Parts	4.406 ± 0.007	7
Plus	4.404± 0.007	6
Plus2	4.41 ± 0.007	6

it is encouraging to see that results are, at least, as good as previously, and maybe marginally better (for Plus, at least); in this particular Mastermind competition, Plus would be the best; however, it is clear that statistics dictate that it could happen otherwise in a different one. At least the maximum number of moves is kept at the same level as Entropy, which might indicate that it achieves the reduction in search space size we were looking for. In fact, the size of the consistent set is practically the same than for Entropy. However, there is some difference in the size of the sets of top scorers, which is shown in Figure 13 and Table 10. Table and graph show that, as intended, Plus and Plus2 reduce the size of the consistent set in the same proportion as Entropy does, but at the same time, since the set of top scorers from which the move is randomly chosen is smaller, the probability of finding the secret code among them is higher; in fact, it is slightly higher than for Most Parts. These two facts, together, explain the small edge in the number of moves, which corresponds also to the small improvement in the secret-playing probability shown in Figure 14. In fact, difference is significant from moves 1 to 3 (using Wilcoxon test) for Plus2 against Most Parts, and from moves 2 and 3 for Plus. Difference in total score is very small between the best (Plus, 57072) and the worst (Entropy, 57189), so, essentially, all methods could obtain the same results. However, we have achieved to (significantly) increase the probability of obtaining the secret combination at each step, which might account for the small difference. Since this difference is offset by other random factors, however, no method is significantly better than other, and the p-value comparing Plus and Entropy is only 0.2780.

As we have seen before, the scenario is different at other sizes, so we will have to experiment them over the $\kappa = 8, \ell = 6$ space. We use the same instance of 5000 combinations as we did in the previous

Table 10. Percentage of times the secret code is among the top scorers for each method, $\kappa = 4, \ell = 6$. Entropy column has been suppressed for clarity.

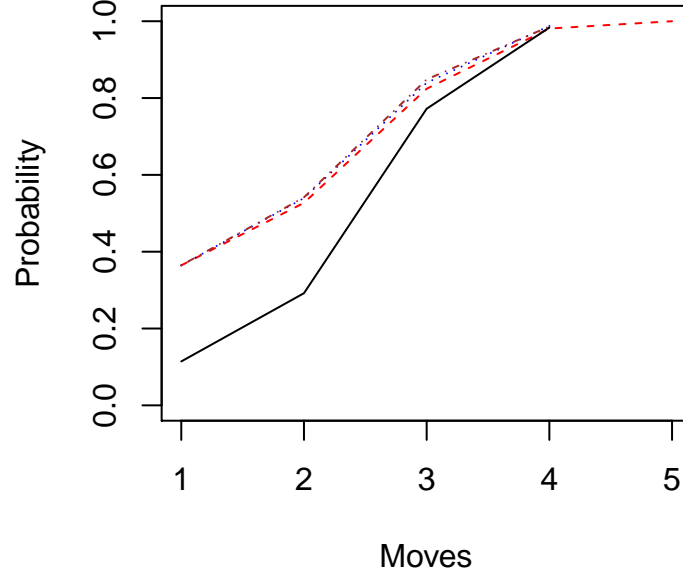| In move #	Plus	Plus2	Most Parts
2	0.364478	0.3644788	0.3644788
3	0.5395962	0.5414171	0.5270987
4	0.8397347	0.8484542	0.8242563
5	0.9877778	0.9862107	0.9810066
6			1

Probability secret code in top scorers

Figure 13. Chance of finding the secret code among the top scorers for the Entropy method (black and solid) and Most Parts (light or red), to which we have added Plus (dotted, blue) and Plus2 (dash-dotted, brown), for $\kappa = 6, \ell = 4$.

Subsection 1.2. This size was big enough to find some differences among methods, and small enough for being able to perform the whole experiment in a reasonable amount of time (around 90 minutes for the whole set). The average number of moves is shown in Table 11. While previously the difference between Most Parts and Entropy was statistically significant, the difference now between Most Parts and Plus2 is significant with p= 0.08536. It is not significant the difference between Plus/Plus2 and Entropy, Plus and Most Parts, and obviously between Plus and Plus2. This is, indeed, an interesting result that shows that we have been able to design a method that statistically is able to beat at least one of the best classical methods, Most Parts, however, the edge obtained by them is not enough to gain a clear victory over both of them.

Let us check whether this difference stems from the design of the new algorithms by looking, as before, at the size of the consistent sets (Fig. 15) and the probability of finding the secret code among the top scorers (Fig. 16). Once again, the difference among the new methods and the old ones is very small, and almost none between them. The average set size is virtually the same as Entropy (98 vs. 99 and 100 after the second move, for instance), but this was to be expected; they are, anyway, smaller than the size of the sets for Most Parts.

The *finding the secret code by chance* probability is, as intended, more similar to Most Parts (dotted and dot-dashed lines over the red dashed line); that is, the probability of finding the secret code among the top scorers is higher than for Entropy, also as intended. This mixed behavior (reduction of search space as in Entropy, probability of finding the secret code among the top scorers as in Most Parts) explain why the methods proposed in this paper reach the results shown in Table 11, which improve, in a
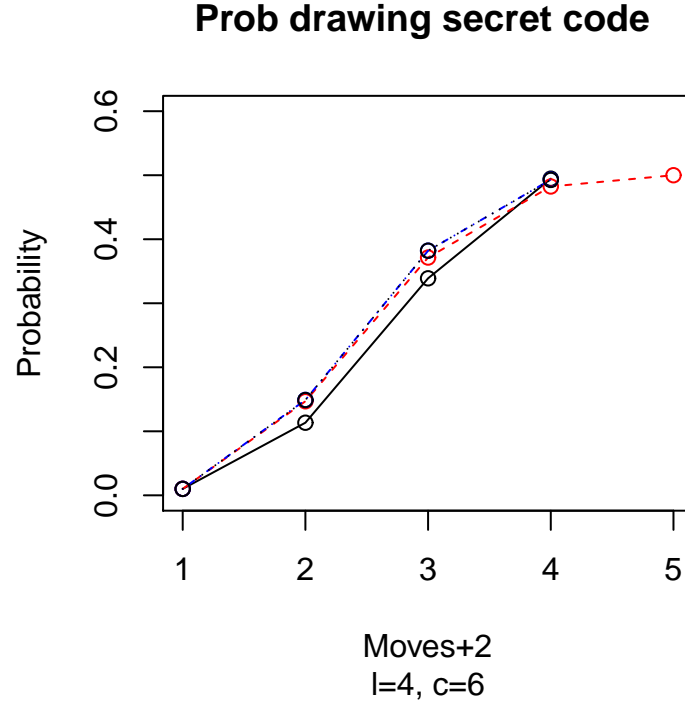
Prob drawing secret code

Figure 14. Chance of drawing the secret code for the Entropy method (black and solid) and Most Parts (light or red) (same as in Figure 6) to which we have added Plus and Plus2, practically one on top of the other and represented with dark-blue and dotted (Plus) and blue dash-dotted (Plus2).

significant way, the state of the art in solutions of the game of MasterMind. But looking more precisely at the actual chance of drawing the secret code (not the binary probability of finding it among the top scorers as in the previous graph), represented in Figure 17 we find the situation very similar to the one shown in 12, however, the probability of drawing the secret code is better for Plus2 than for Most Parts, and significantly so for all moves up to the 5th. However, it is still worse for Plus2 than for Entropy for the third move ($x = 2$), which accounts for its eventual tie with it. However, the difference with respect to Most Parts seems to be enough to achieve a small difference at the end.

As a conclusion to this section, we have proposed two new methods for solving mastermind that try to improve the capability of drawing the secret combination at the beginning of the game while, at the same time, decreasing the size of the search space as Entropy does. The proposed methods obtain results that are better than the worst of the previous methods (Entropy or Most Parts), but not significantly better than the best method (Most Parts or Entropy). However, they are robust in the sense than they are at least as good (statistically) as the best for each size, so in a sense it could be said that we have achieved a certain degree of success. Let us see if this conclusion holds by slightly changing the circumstances of the game by using a different initial combination.

Table 11. Average and maximum number of moves for the two new search strategies, Plus and Plus2, along with the previously shown Most Parts and Entropy for $\kappa = 8, \ell = 4$.

Method	Number of moves		Median
	Average	Maximum	
Method	Number of moves: Average	Number of moves: Maximum	
Entropy	5.132 ± 0.012	8	5
Most Parts	5.167 ± 0.012	8	5
Plus	5.154 ± 0.012	8	5
Plus2	5.139 ± 0.012	8	5

Remaining combinations, average size
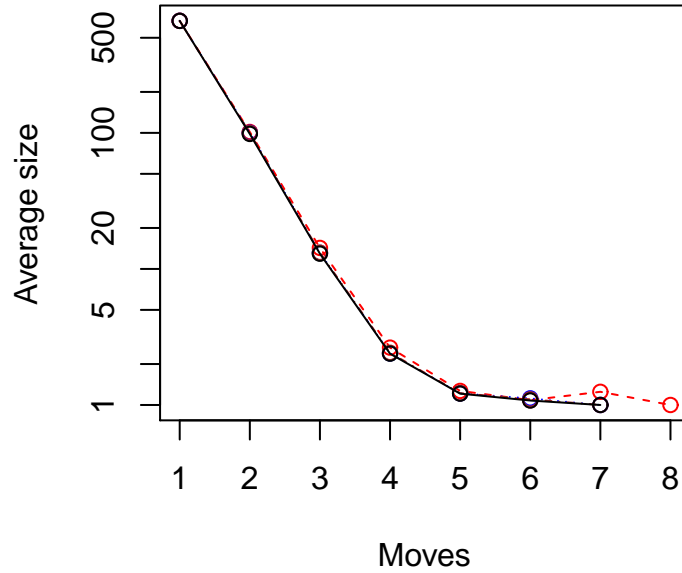
Figure 15. Average size of the consistent set, that is, the set of solutions that have not been discarded at a point in the game for $\kappa = 8, \ell = 4$. As usual, red or light dashed line represents Most Parts and solid black Entropy, dotted and blue for Plus and dash-dotted and brown for Plus2. Please note that the y axis is logarithmic; we have put it that way to highlight differences.

Studying the effect of a different starting combination on the differences among algorithms

To test whether, under different circumstances, the newly proposed methods perform as well as the best one, and also check the influence of changing the initial combination to a different interpretation of Knuth's first move, we will compare the two best algorithms seen before, Plus2 and Entropy for $\kappa = 8$ using ABCA as first move (instead of the previously used ABCD). In principle, if we apply the partition

Prob. secret code in top scorers

Figure 16. Chance of finding the secret code among the top scorers for the Plus (dotted, blue), Plus2 (dash-dotted, brown), Entropy method (black and solid) and Most Parts (light or red).

score to the first move (which can be done, even in the absence of a reduction of search space brought by a move) a combination with different symbols (such as ABCD) obtains the maximum entropy encore. However, in the absence of information about the secret combination it is again a empirical exercise to test different initial moves, as has been done in [20], for instance.

Table 12. Comparison of average and maximum number of moves using ABCA as first move and the best of the previous ones.

Method	Number of moves		
	Average	Maximum	Median
Most Parts	5.167 ± 0.012	8	5
Entropy (ABCD)	5.132 ± 0.012	8	5
Plus2 (ABCD)	5.139 ± 0.012	8	5
Entropy (ABCA)	5.124 ± 0.012	8	5
Plus2 (ABCA)	5.116 ± 0.012	8	5

The summary of the number of moves is again shown in Table 12. *A priori*, the average number of moves is better than before. However, the only significant difference (as usual, using paired Wilcoxon test) is between Entropy(ABCA) and Plus2(ABCA) and Most Parts and Plus. The difference between the ABCA and ABCD versions of both algorithms is not significant. The difference between Plus2 and Entropy using ABCA as first move is, once again, not significant, proving that the new algorithm
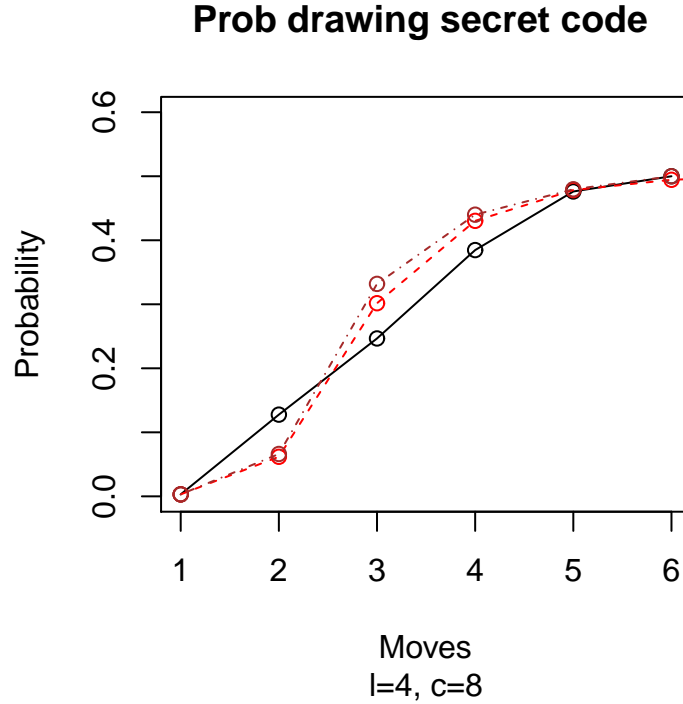
Prob drawing secret code

Figure 17. Chance of finding the secret code among the top scorers for the Entropy method (black and solid) and Plus2 (light or brown, dot-dashed). Plus has been suppressed for clarity; the x axis is shifted by one, with $x = 1$ representing the second move.

proposed, plus2, is as good as the best previous algorithm available for the size (which, in this case, is Entropy).

We can also conclude from this experiment that, even if there is not a significant difference for a particular algorithm in using ABCD or ABCA, it is true that results using ABCA are significantly better for Entropy and Plus2 than for other algorithms such as Most Parts, with which there was no significant difference using ABCD. We have not tested Most Parts and Plus in this section since our objective was mainly comparing the best methods with a new starting move with all methods using the other move; however, we can more or less safely assume that results will be slightly, but not significantly, better, and that they will be statistically similar to those obtained by Entropy and Plus2.

It is clear that, in this case, the advantage is mainly due to the changes in search space from the first move, however, this will have an influence on the probability of playing the secret combination, as shown in Figure 18. In both cases (Entropy and Plus2) the probability is *lower*, however, the result is (not significantly) better. The differences must be due, then, mainly to the difference in consistent set size, which is shown in Table 13, but these differences are not so clear-cut as should be expected, that is, smaller sizes throughout all moves. The size is actually bigger for ABCA in the first move, although smaller in the second. These two will actually have little influence on the outcome (since, at that stage in games, drawing the secret code is mainly the product of the scoring algorithm and the *composition* of the top scorers). However, there is a significant difference before move 5th, which is when in some cases the consistent set is reduced to one, which is also reflected in the differences in score represented in Figure 19. Up to the fifth move (label 2 in the probability graph 18), Entropy(ABCD) is better (negative difference),
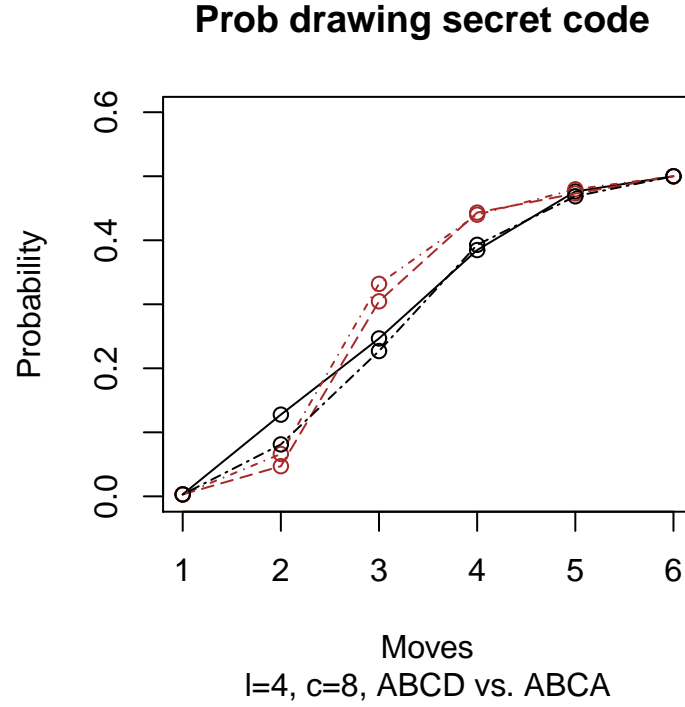
Prob drawing secret code

l=4, c=8, ABCD vs. ABCA

Figure 18. Chance of finding the secret code among the top scorers for the Entropy method (black and solid) and Plus2 (light or brown, dot-dashed), both for ABCD as first move, and Entropy (black, dot-long-dash) and Plus2 (brown or light, long dash) for ABCA .

Table 13. Average and standard deviation of the number of combinations remaining after every move for Entropy ABCD and ABCA (the quantities for Plus2 are practically the same). The number of combinations is the same after the first move (not shown here), since both are playing the same first move.

Before move #	Entropy ABCD	Entropy ABCA
2	666 ± 310	706 ± 327
3	101 ± 67	99 ± 69
4	13 ± 10	13 ± 10
5	2.4 ± 1.5	2.28 ± 1.48
6	1.21 ± 0.46	1.23 ± 0.5
7	1.09 ± 0.29	1.19 ± 0.39

Entropy(ABCD) < Entropy(ABCA). However, it accumulates score (remember that higher score is worse) in the 6th move, which eventually implies the victory (which we should note is not significant) of Entropy (ABCA).

The improvements brough by the new Plus2 algorithm and the change in initial should be highlighted if we compare significantly different methods: Most Parts vs. Plus2(ABCA). The difference in consistent set size is shown in 20, and it shows the same pattern as the previous example 13, only it is quite clear now that there is a consistent reduction from the first move. It is interesting to note again that the
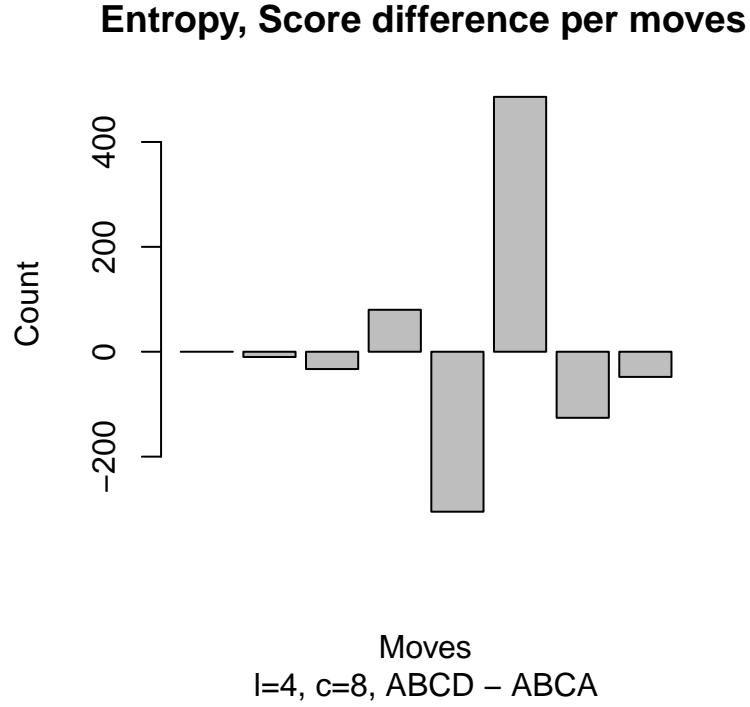
Entropy, Score difference per moves

Figure 19. Difference in score among the number of games won by Entropy (ABCD) and Entropy (ABCA) in x moves for $\kappa = 8, \ell = 4$. Method is the same as for figure 3, that is, product of difference and number of moves.

reduction in search space (or consistent set) size provoked by using ABCA as first move takes place in the *second* move, not the first move. This would already be a harbinger of a better game playing, but by looking also at the Figure 21 representing the probability of playing the secret combination we see that it is higher from the 4th move (label 3) and is kept that way until the end. the difference in score achieved is shown in Figure 22 which shows clearly first that the advantage of Plus2 is settled at the latest moves; Plus2 is able to play the winning combination lest often than Most Parts (1st to 5th move), but this might be due to the fact that Plus2 wins more games (and thus has a higher score), while out of the 5000 games, there are more left to finish when Most Parts (ABCD) arrives to the 5th game and is thus beated (significantly, as indicated above) by Plus2. Please note also the clear phase shift from the 5th to the 6th game, when Most Parts changes from having a better score than Plus2 to having a worse score, which is the final result. Most of the difference can be assigned (if such a thing is actually possible, since it is obvious that victory or defeat are forged over the whole range of moves) to the 7th move, when Plus2 (ABCA) has already finished a big percentage of the games, but Most Parts (ABCD) has not.

At any rate, in this section we have proved that the two mechanisms we studied in the previous sections is also at work to provoke the advantage of one algorithm over other, and that studying mainly the size of consistent sets and the probability of drawing the secret code, and when one mechanism for finishing the game takes over the other, is the best way of evaluating different algorithms for playing the game of Mastermind.

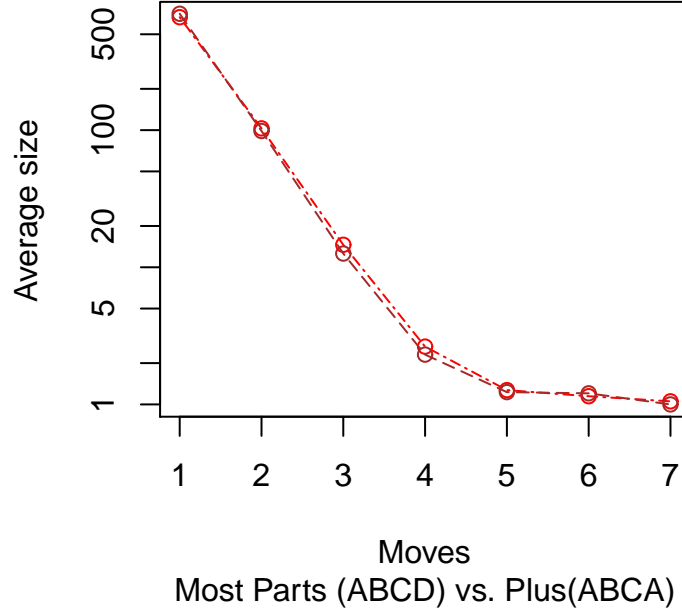## Remaining combinations, average size

Figure 20. Average size of the consistent set, that is, the set of solutions that have not been discarded at a point in the game for $\kappa = 8, \ell = 4$. As usual, red or light dashed line represents Most Parts and two-dashed and brown for Plus2. Please note that the y axis is logarithmic; we have put it that way to highlight differences.

Conclusions and discussion

In this work we have analyzed the solutions to the game of Mastermind in a novel way, taking into account not only the reduction of the size of the search space brought by the combination played, but also how every method scores the components of the search space and whether the secret combination is among the top scorers, that is, the actual probability of drawing the secret combination at each move. Using always combinations of length equal to 4, we have proved that the fact that Most Parts has an increased chance of finding the secret combination among its top scorers counterbalances the effective reduction of the set of consistent combinations brought by Entropy for six colors; the difference among the probabilities for both method decreases with the search space size (which we have proved for $\kappa = 8$), making Entropy beat Most Parts for that configuration.

Using that fact, we have proposed two new methods that effectively combine Entropy and Most Parts by reducing search space size as the former, and having a set of top scorers to choose at worst as bad as Most Parts. These two methods, Plus and Plus2, behave statistically in the same way for both sizes tested, are marginally (not significantly) better than Most Parts for $\kappa = 6, \ell = 4$, and significantly better than Most Parts and marginally better than Entropy for $\kappa = 8, \ell = 4$, this method being itself only marginally better than Most Parts for that size. Besides, results obtained by Plus/Plus2 are an improvement over the state of the art, published, for instance, in [**?**] (which is comparable to that obtained by Entropy and, thus, probably not statistically significant).
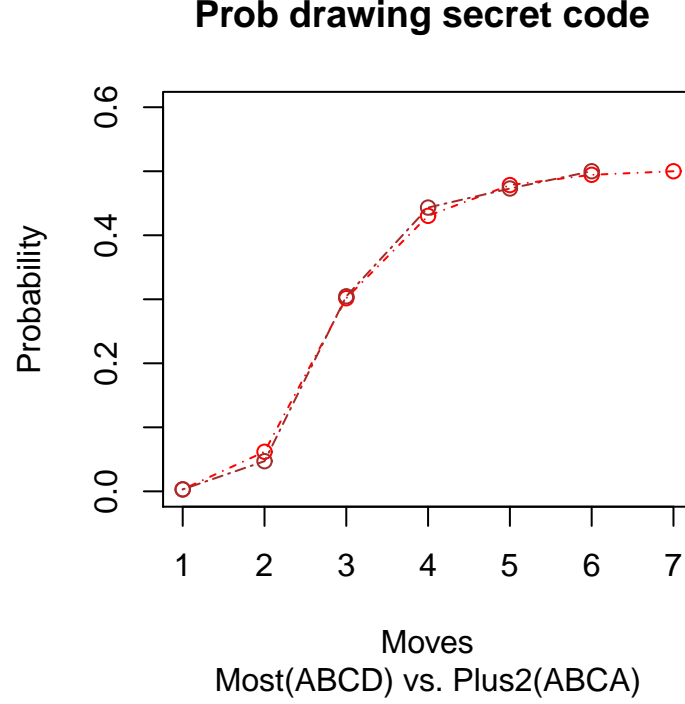
Prob drawing secret code

Figure 21. Chance of drawing the secret code for the Most Parts (ABCD) method (red or light and dot-dashed) and Plus2 (ABCA) (light or brown, two-dashed).

The two new methods presented do not present statistically significant differences. This could be interpreted by stating that combining a scoring strategy that reduces search space *and* gives top scores to the hidden combination consistently, in general, is the most profitably course of action. At the same time, doing to in different ways will not lead to significant differences. However, although it is tempting to generalize this to spaces of higher dimensions, heuristically it cannot be done. We can, however, affirm that combining several scoring strategies, specially Entropy and Most Parts, will yield better results even if using samples of the whole consistent set. Besides, both methods, but specially Plus2, is always (for the two sizes tested, which is considered enough) as good as the best previous method described, which makes it more robust.

This paper also introduces and tests a methodology for testing different algorithms for solving Mastermind. While the first approximation would be to empirically measure the average number of moves, and the first would design methods that maximize the reduction of search space every move, we have introduced here the measurement of the probability of drawing the secret code as a third empirical test to consider when designing a new method, since the success of a solution depends, for the first moves, in that probability, to be followed by the reduction of search space to a single element, as considered so far.

In theory, these results and methodology could be extended to spaces with bigger sizes; however, in practice, the increase in time complexity of the algorithm bars it except from the simplest extensions. While solving $\kappa = 6, \ell = 4$ takes around 0.5s, $\kappa = 8, \ell = 4$, whose search space is only 4 times as big, needs around 10 times as much time, around 5 seconds. One of the bottlenecks of the algorithm is the need to compare all elements in the consistent set with each other, which is approximately quadratic in size; this needs to be done for each step, besides, the number of steps also increases in a complex way with
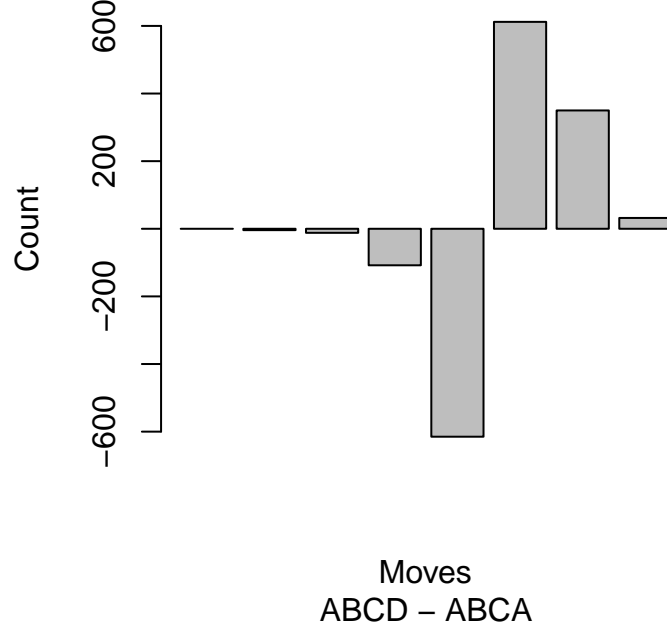
Most Parts – Plus2, score difference per mc

Figure 22. Difference in score among the number of games won by Most Parts (ABCD) and Plus2 (ABCA) in x moves for $\kappa = 8, \ell = 4$. Method is the same as for figure 3, that is, product of difference and number of moves.

the increase in search space size. However, in time it will become possible to test in a reasonable amount of time whether these new methods are still better than the classical ones, and in which proportion. It is impossible to know in advance what will be the influence of finding the secret code among the top scorers; in fact, it *increases* from $\kappa = 6$ to $\kappa = 8$: the size of the consistent set increases, but the probability of finding the secret code among them also increases (although the actual probability of drawing that code decreases too); however, the actual size of that set will also have an influence, with bigger sizes decreasing the actual chance of playing the secret combination. How the three quantities will change with the problem size (and, actually, with both dimensions of the problem: number of colors κ and combination length ℓ) is beyond the scope of this paper, and might not be easy to compute analytically.

It is more interesting, however, to use these results for methods that use a sample of the set of consistent combinations, such as evolutionary algorithms [?, 20]. Results obtained here can be used in two different ways: by taking into account the size of the consistent set to set the sample size, which was set heuristically [?] or fixed [20], and also by using a combination of Most Parts and Entropy scores to compute the fitness of a particular combination. This will allow to find solutions to Mastermind problems in bigger search spaces. However, the limiting factor keeps on being the size of the consistent set that will be used to score the combinations; since sample size increases with problem size, eventually a non-feasible limit (in term of time or memory usage) will be reached. However, using even a small size will extend the range of problems with feasible solutions. Finding a way to score solutions that is less-than-quadratic in time will also extend that range, although another avenue, already explored years ago [?], would be to play non-consistent combinations in some cases if enough time without finding consistent combinations

passes.

This leaves as future work improving the scoring used in evolutionary algorithms using these results, and checking how far these solutions will go into the search space. Managing to find a solution to the $\kappa = 12, \ell = 8$ in under one hour would be a good target, but it will need a good adjustment of the evolutionary algorithm parameters, as well as tweaking the implementation as far as it will allow (because implementation matters [?]). Using either a distributed computing environment such as the Evolvable Agents [?] or SofEA [?] will probably be needed to in order to shorten total time to solution. However, an parallel version is not trivial, since things such as the consistent set might have to be centralized.

Acknowledgments

References

1. Weisstein EW. Mastermind. From MathWorld–A Wolfram Web Resource. URL `http://mathworld.wolfram.com/Mastermind.html`.

2. Wikipedia (2009). Mastermind (board game) — Wikipedia, The Free Encyclopedia. URL `http://sl.ugr.es/001X`.

3. Francis J. Strategies for playing MOO, or "Bulls and Cows". `http://www.jfwaf.com/Bulls%20and%20Cows.pdf`.

4. Chen ST, Lin SS, Huang LT (2007) A two-phase optimization algorithm for mastermind. Computer Journal 50: 435-443.

5. Asuncion A, Goodrich M (2010) Nonadaptive mastermind algorithms for string and vector databases, with case studies. Knowledge and Data Engineering, IEEE Transactions on : 1–1.

6. Goodrich MT (2009) The mastermind attack on genomic data. CoRR abs/0904.4458.

7. Focardi R, Luccio F (2010) Cracking bank pins by playing mastermind. In: Boldi P, Gargano L, editors, Fun with Algorithms, Berlin Heidelberg: Springer-Verlag, volume 6099 of *Lecture Notes in Computer Science*. pp. 202-213.

8. Focardi R, Luccio F (2011) Guessing bank pins by winning a mastermind game. Theory of Computing Systems : 1–20.

9. Goodrich M (2009) On the algorithmic complexity of the Mastermind game with black-peg results. Information Processing Letters 109: 675–678.

10. Gagneur J, Elze M, Tresch A (2011) Selective phenotyping, entropy reduction, and the mastermind game. BMC bioinformatics 12: 406.

11. Boldi P, Gargano L (2012) Special issue on fun with algorithms. Theory of Computing Systems : 1–2.

12. Several Authors. How to solve the "Mastermind" guessing game? `http://stackoverflow.com/questions/1185634/how-to-solve-the-mastermind-guessing-game`. Question posed in StackOverflow and many answers.

13. O´Geran J, Wynn H, Zhigljavsky A (1993) Mastermind as a test-bed for search algorithms. Chance 6: 31–37.

14. Stuckman J, Zhang GQ (2006) Mastermind is NP-complete. INFOCOMP J Comput Sci 5: 25–28.

15. Kendall G, Parkes A, Spoerer K (2008) A survey of NP-complete puzzles. ICGA Journal 31: 13-34.

16. Viglietta G (2011) Hardness of Mastermind. CoRR abs/1111.6922.

17. Kooi B (2005) Yet another Mastermind strategy. ICGA Journal 28: 13-20.

18. Doerr B, Spöhel R, Thomas H, Winzen C (2012) Playing Mastermind with Many Colors. ArXiv e-prints .

19. Knuth DE (1976-77) The computer as Master Mind. J Recreational Mathematics 9: 1–6.

20. Berghman L, Goossens D, Leus R (2009) Efficient solutions for Mastermind using genetic algorithms. Computers and Operations Research 36: 1880-1885.

21. Irving RW (1978-79) Towards an optimum Mastermind strategy. Journal of Recreational Mathematics 11: 81–87.

22. Neuwirth E (1982) Some strategies for Mastermind. Zeitschrift fur Operations Research Serie B 26: B257–B278.

23. Bestavros A, Belal A (1986) Mastermind, a game of diagnosis strategies. Bulletin of the Faculty of Engineering, Alexandria University .

Figure Legends

Tables