university of groningen / faculty of science and engineering

Solving "Are you the one?" game with entropy based decisions

Bachelor's Project Thesis

N.S.N. Bosgoed, n.s.n.bosgoed@student.rug.nl
Supervisor: Dr H.A. de Weerd

Abstract: The reality TV show "Are you the one?" is a game show loosely based on the game Mastermind. Mastermind is a 2-player 6-colour code cracking game. The purpose of this project is to produce Python-based programs which solve the "Are you the one?" game for an n-digit code within n steps, with n a number between 1 and 10 inclusive. The programs apply an algorithm which chooses the best move based on highest entropy. One program shows that it is always possible to solve the game in n steps up to 7 couples. The other program shows that it is most likely to solve the game in n steps up to 10 couples.

1 Introduction

If one thinks about reality shows on TV, one might think that such a show does not contain any logic or mathematical background. However, a social experiment reality television dating show called "Are you the one?"* contains some traces of a game called Mastermind in it. In this dating show 10 young single women and 10 young single men must find their perfect match in 10 rounds. The correct matches are determined beforehand by experts through in-depth interviews, questionnaires, and compatibility testing. Each round the contestants must couple up the 10 men with the 10 women. This coupling up is called the "matchup ceremony". The show lets them know how many of these matches are correct through the lighting of an amount of beams. As determining the correct way to match 10 couples is fairly difficult, an extra round is introduced: the truth booth. Before each matchup ceremony with 10 couples, one single couple can be sent into the truth booth. For this couple, it will be revealed whether they are a 'perfect' match or not. If they are a match, they do no longer need to switch and the problem of matching up the men and women has one uncertain couple fewer.

The game outlined in the previous paragraph has some interesting similarities with Mastermind. Mastermind is a 2-player code cracking game. One player starts the game by making a code with 4 tokens which each contains one of 6 possible colours. These tokens may be the same colour. The opponent has 10 rounds to guess the code. Each round the opponent can guess a code and the first player responds with black and white pegs. Each white peg indicates a correct colour at the correct position and each black peg indicates a correct colour at an incorrect position. If a token is not a token of a correct colour, one of the response positions will be empty. The minimum number of these pegs is 0 and the maximum is 4. The gameboard of Mastermind is shown in Figure 1.1.

The case of "Are you the one?" occurs when the number of coloured tokens would be 10, the amount of colours would be 10 and each colour may only occur once. The number of rounds is still 10. The men denote the places of the tokens and the women denote the colours of the tokens. Only the white pegs, denoted by light beams, play a role in this TV show. Since each colour is used exactly once, the white pegs would be complemented with black pegs to a total of 10. This means that the number of black pegs is not important, because without them the empty positions would indicate the exact same, and therefore the black pegs are omitted. The truth booth round can be seen as an extra

*https://www.mtv.com/shows/are-you-the-one

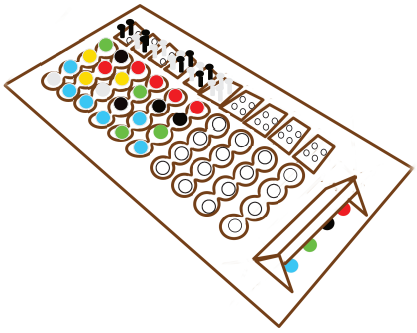

Figure 1.1: The game board of Mastermind. At the top on the left the guessed codes are shown with on the right the response pegs. At the bottom under the hood the code to guess is shown.

inbetween round where only one colour is tried at one of the positions. If it is correct there would be one white peg, if it is incorrect there would be no pegs. The adjusted game board is shown in Figure 1.2

Solutions for Mastermind itself, to produce an optimal strategy which requires the minimum amount of rounds, go back to the seventies. Knuth [1977], a computer scientist, used the strategy to choose the candidate code which would result in the minimum set of possible codes after a round. Neuwirth [1982], a statistician, researched One-Step-Ahead Expectation Minimizing-Strategies, where entropy measured "goodness" of a partition of data for information. Currently there are still several fields that examine algorithms to find the best strategy to play Mastermind. Examples of fields are Machine Learning and Reinforcement Learning. In Machine Learning are for instance Genetic Algorithms, cluster-based algorithms and population-based algorithms elaborated in the papers of Berghman et al. [2009], Partynski [2014] and Ugurdag et al. [2006]. Some researchers nowadays have pointed out the potential usefulness of entropy, just like Neuwirth, in solving Mastermind [Merelo-Guervós et al., 2013]. At the moment Mastermind still does not have a perfect solution. In the course of time, all these strategies have in common that they use traditional Mastermind. The difference between the research methodologies is that some methodologies only use the 4-colour code whereas other methodologies use longer colour codes.

However, in this research it is not about Mastermind, but about the game show "Are you the one?". As stated in the previous paragraphs, this is an interesting variation on Mastermind where the candidates go on dates to gather extra information. This will help them in the matchup ceremony because the matchmaking algorithm is based on everyone's personal value. However, it may be possible to determine the correct match-up without use of this external information. It would be interesting to see whether a pure entropy-based search algorithm would be able to solve "Are you the one" within the limits outlined in the game show. In this paper, it is therefore investigated whether it is possible to uncover the correct match-up of n couples using at most n rounds of matchup ceremonies and truth booths without external information. Here the n is up to 10 couples since the TV show has set that amount as level of difficulty.

To uncover this, two Python based programs are used with the Numpy package. Through entropy it is determined which action would be best in each step. All possible combinations of codes are produced with permutation. The outcome will be the maximum amount of rounds needed for each number of couples through working out a game tree and through playing the game a 1000 times with random solutions.

In the section Methods it will be explained how the programs work in detail. In the section Results, it will be elaborated how many rounds were needed to solve the game for random solutions and the maximum amount of rounds needed through computations of game trees. The last section Discussion will explain the performance and further research.

2 Methods

As mentioned in the introduction, the matches for a 10 couple game can be seen as a 10-colour code. However, for a program it is more logical to use digits instead of colours. So, a matchup can be represented as a n-digit code that includes each of the digits from 0 to n-1 exactly once. An example

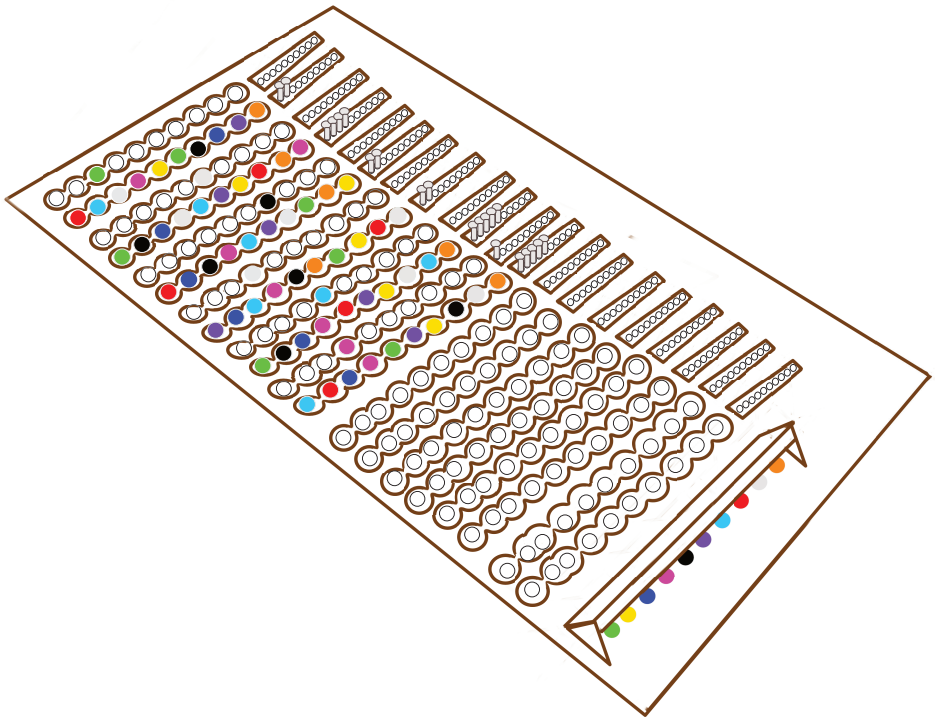

Figure 1.2: The adjusted game board of Mastermind to show the "Are you the one" variation. At the top on the left the guessed codes are shown with on the right the response pegs. At the bottom under the hood the code to guess is shown.

of a possible solution would be: "0 1 2 3 4 5 6 7 8 9". In this instance, man 0 would be coupled with the woman 0, man 1 with woman 1, and so forth. To get other solutions the digits need to be switched. The truth booth codes consist of -1's and one digit; this digit and its place corresponds to the couple. For example: "-1 -1 -1 -1 -1 -1 4 -1 -1 -1" means that man 6 and woman 4 are asked in the truth booth in a 10 couple game. Since the truth booth does not entail all couples but only one, only a matchup ceremony can end the game. The contestants have solved the game if they manage to get all beams lit up during a matchup ceremony.

2.1 By hand

To get more insight a game tree can be used. The tree in Figure 2.1 represents the actions that should be taken in order to solve "Are you the one" in the minimal number of rounds. One can see that, for a game of 3 persons, every round in the game has either 3 possible outcomes (in case of a matchup ceremony) or 2 possible outcomes (in case of a truth booth). For 10 persons that would be 10 or 2 possible outcomes. In the worst case scenario, contestants would be able to solve the game (i.e. get 3 beams in a match-up) in the second round of play (i.e. at the second iteration of truth booth and match-up). The tree excludes cases wherein no strategy would be used and illogical decisions, for example a couple that is already confirmed not to be a match, would be entered. Nevertheless, one can imagine, the game tree with all logical, still strategical options would have a lot more branches. A worked out version by hand would take far too long and would be prone to many duplications and errors.

To show how immense the game tree can grow, the amount of possible solutions can be calculated with permutations. At the start of the original game there are $10! = 3\,628\,800$ permutations

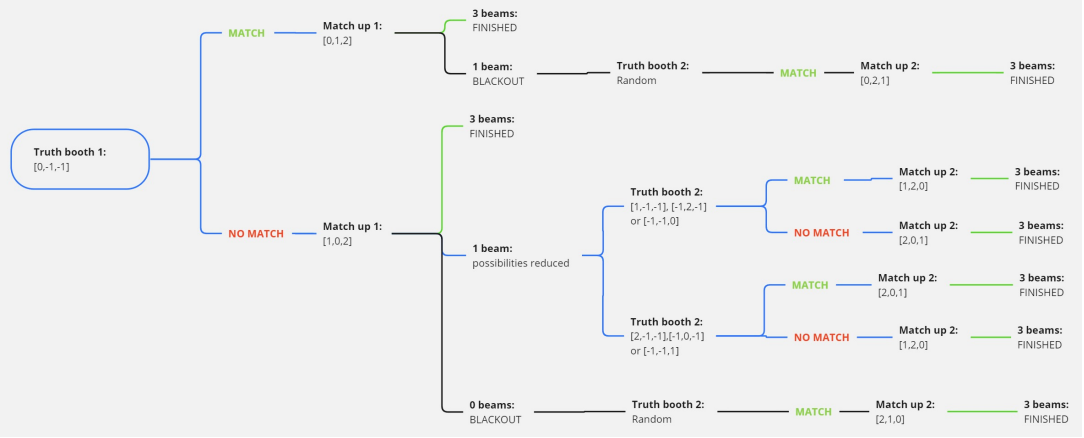

Figure 2.1: A game tree for "Are you the one?" with 3 couples made in Miro. The tree shows that the correct matchup is always found within 2 rounds. This means that for 3 couples the game is solved within 3 rounds.

of possibilities of matches. This means that a complete tree that solves the game for any possible code would therefore also have 3 628 800 leaf nodes. Each round should eliminate as many of those possibilities as possible. As we have no influence on the result of a truth booth or a matchup, the possible results should be split as equally as possible. This results in the best worst-case performance. After all, if the results are not split equally, there is a chance that the result with the most remaining options could be hit. More options left means that more eliminations are needed. More necessary eliminations leads to more remaining rounds, which is unwanted.

2.2 Entropy for decision making

Each round, whether it is a truth booth or a matchup ceremony, a decision has to be taken which option is most ideal. For choosing the right code to enter the entropy of Shannon is deployed. This criterion calculates how well each group is divided into subgroups. In this subsection firstly the intuition behind the method will be outlined to explain why it can be applied to "Are you the one?". To round off this subsection the mathematical implementation of the entropy measure will be elaborated.

Serano [2017] thought up an idea that it can be explained with the analogy of buckets and balls. Suppose all the possible solutions left in the game are balls. For each decision (i.e. a matchup or a truth booth), a set of n or 2 buckets can be filled. Each bucket is filled for each possible outcome. For example, in a game with 10 couples, each matchup result in the remaining balls are distributed over 10 buckets, each representing the number of beams that are lit for that particular decision. For example, if contestants choose to enter the code "0 1 2 3 4 5 6 7 8 9", the ball "0 1 2 3 4 5 6 7 9 8" would be placed in the bucket corresponding to 8 beams, as 8 of the numbers in the entered code match the code on the ball. Once the outcome has been observed (e.g. 8 beams are lit), only the corresponding bucket remains. The balls in the other buckets are discarded. The best outcome is the least amount of balls going to the next round, since the solution has to be filtered out. However, the worst scenario is the highest amount of balls going to the next round, which will lead to more rounds. Since the goal is to solve the game for any possible code, the decision that results in the most even distribution of balls across buckets is the best option.

So, for each set of buckets it is calculated how well the outcomes differ from each other. If the outcomes are mostly similar, for example if for ten couples they nearly all result in 2 beams being

lit up, a few in 3 beams, and one in 10 beams, the entropy will be low. If the outcomes are more different, the entropy will be higher. In the first case the best case scenario would be if 10 beams are lit, since there would be only one solution left. However, this also means that if there are 2 beams lit, almost all solutions will be left. In the case with a more even distribution of balls across outcomes, there is no chance that as many solutions as in the first case, and the worst case scenario, will be left. For this reason the set of buckets with the highest entropy will be chosen and the corresponding decision will be inserted. The beams corresponding to that decision will be lit and the balls corresponding to that amount are deployed for the next round as remaining possible solutions. For each possible solution the set of buckets will be filled again with those balls. When there is only one ball left, the correct matchup has been found. Since the solution is selected (in the programs random, in the TV show by experts) unknowingly by the contestants to be one out of the 3 628 800 possible options for 10 couples, there is no influence on the amount of beams that are lit. The purpose of the game is to find the solution for n couples within n rounds, no matter what the solution is.

The intuition described above can be described mathematically in terms of entropy. The entropy for the Shannon criterion is calculated with the following function:

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x)$$

P(x) is in this function the proportion of solutions with x beams in the case of a matchup ceremony or the proportion of perfect match (1) versus not a match (0) in case of a truth booth within the set X. X is the set of possible outcomes (number of beams in match-up and yes/no in truth booth) at the moment. The highest entropy indicates the best spread of results. This entropy is calculated for each possible course of action and since the best spread is wanted, the option with the highest entropy is chosen as decision.

An example of this entropy is if there are only 3 couples and it is the start of the game; all remaining options are “0 1 2”, “0 2 1”, “1 0 2”, “1 2 0”, “2 0 1” and “2 1 0”. The calculation for the proportions of the choice of first truth booth “0 -1 -1” would be:
$P(1) = \frac{2}{6} = \frac{1}{3}$,
since 2 out of 6 solutions (“0 1 2” and “0 2 1”) would result in a perfect match outcome for the given truth booth.
$P(0) = \frac{4}{6} = \frac{2}{3}$,
since 4 out of 6 solutions (“1 0 2”, “1 2 0”, “2 0 1” and “2 1 0”) would have no match as consequence. The calculation for the entropy is as follows:

$$\begin{aligned} & H(\{\text{“012”}, ..., \text{“210”}\}) \\ &= \sum_{x \in \{0,1\}} P(x) \log_2 \frac{1}{P(x)} \\ &= \frac{1}{3} \log_2 \frac{1}{\frac{1}{3}} + \frac{2}{3} \log_2 \frac{1}{\frac{2}{3}} \\ &= \frac{1}{3} \log_2 3 + \frac{2}{3} \log_2 \frac{3}{2} \\ &\approx 0.918 \end{aligned}$$

In the programs the entropy is produced with the package pyinform.blockentropy[†].

2.3 The programs

To calculate how many rounds are needed to solve the game for a random solution two programs are written. One program called approach M(aximum) is written as a game tree, working out all possible ways a game can occur. The other program is a game itself, denoted as approach L(ikelihood). The game tree in approach M calculates all possibilities to determine the maximum amount of rounds needed, which is a worst case scenario. By playing the game in approach L, samples can be taken to determine the likelihood of this worst case scenario happening. In both programs entropy is applied to make decisions. Due to the large amount of calculations for the program of approach L, this program is split into two parts. The first part is data that can be calculated beforehand, the *datamaking*, and the second part is the game itself.

[†]PyInform, `https://elife-asu.github.io/PyInform/`

2.3.1 Approach Maximum

As previously stated the game tree in approach Maximum works out all possible ways a game can occur. For each step of the game tree the best option to take is calculated with entropy. When that best option is decided, all the codes are split into groups where the same result due to that option denotes the same group. This means that for the truth booth round they are split into two groups of no match or perfect match. For the matchup round they are split into the groups 0,1,2,..,n-2 and n beams with n the amount of couples. Each of those groups represents a branch. For each of those branches the best solution is calculated again with entropy and split into consequences again. The depth of the game tree built is the maximum amount of truth booths and matchup ceremonies (for example, 3 truth booths and 2 matchup ceremonies would accumulate to a depth of 5). For 3 couples the resulting game tree is pictured in Figure 2.2 .

2.3.2 Approach Likelihood: Data making

The first part of the game in the approach Likelihood is the making of two databases: the *datamaking*. The rows of the database are the possible actions that can be taken, and are deployed in the game itself to calculate which action has the highest entropy. These databases are constructed in numba with the usage of NumPy (imported as np). Note that in the program of the game the -1 is not applied, but a 0 and the couples start at 1 instead of 0.
First, an array of all possible matchups, *possible_matchups* which is abbreviated as *pm*, is needed. The first array $a = [1, 2, ..., n]$ is the only line changed manually in the program each time to produce the databases with the needed amount of couples. The length is calculated to produce other arrays with the same range. The list is extended with multiset_permutations from sympy.utilities.iterables [‡] and concatenated (np.concatenate) to the array as described in Algorithm 2.1. The array *pm* must start with a beginvalue, because numba will otherwise not compile. Therefore the array *a* is first added and later deleted (np.delete) to avoid duplicates.

Algorithm 2.1 Add all permutations to array *pm* (*possible_matchups*)

$a \Leftarrow [1, 2, ..., 10]$
$n \Leftarrow$len(a)
$pm \Leftarrow [a]$
for p in multiset_permutations(a) **do**
  $pm \Leftarrow pm$ with $[p]$ concatenated to pm at axis 0
end for
$pm \Leftarrow pm$ with the deleted entry on (0, 0)

Algorithm 2.2 Add all truth booths to array *pt* (*possible_truthbooths*)

$pt \Leftarrow n \times n$ matrix filled with zeros
$pt \Leftarrow pt$ with on the diagonal ones
for x from 2 to n+1 **do**
  $temp \Leftarrow n \times n$ matrix filled with zeros
  $temp \Leftarrow temp$ with on the diagonal x's
  $pt \Leftarrow pt$ with $temp$ concatenated to pt at axis 0
end for

Secondly, an array of all possible truth booths, *possible_truthbooths* which is abbreviated as *pt*, is produced. The array *pt* must be filled before the loop, therefore the array starts with an n by n array filled with zero's and with 1's on diagonal representing all the possible truth booths for female 1. In each loop of the for loop the truth booth of the next female is added. These actions are executed through NumPy standard functions np.zeros and np.fill_diagonal as can be seen in Algorithm 2.2.

As the arrays are complete, the databases can be produced. The first database is a matrix of all possible matchups versus all possible matchups, so *pm* versus *pm*. The first possible matchups indicates the possible options. The second possible matchups indicates all the possible solutions. The elements of the matrix are the amount of beams which can result in the solution indicated, which is computed by the function sum_all_jit [§] with one

[‡] https://docs.sympy.org/latest/modules/utilities/iterables.html

[§] copied from the sum_all in: http://numba.pydata.org/numba-doc/0.12/tutorial_numpy_and_numba.html?external_link=true

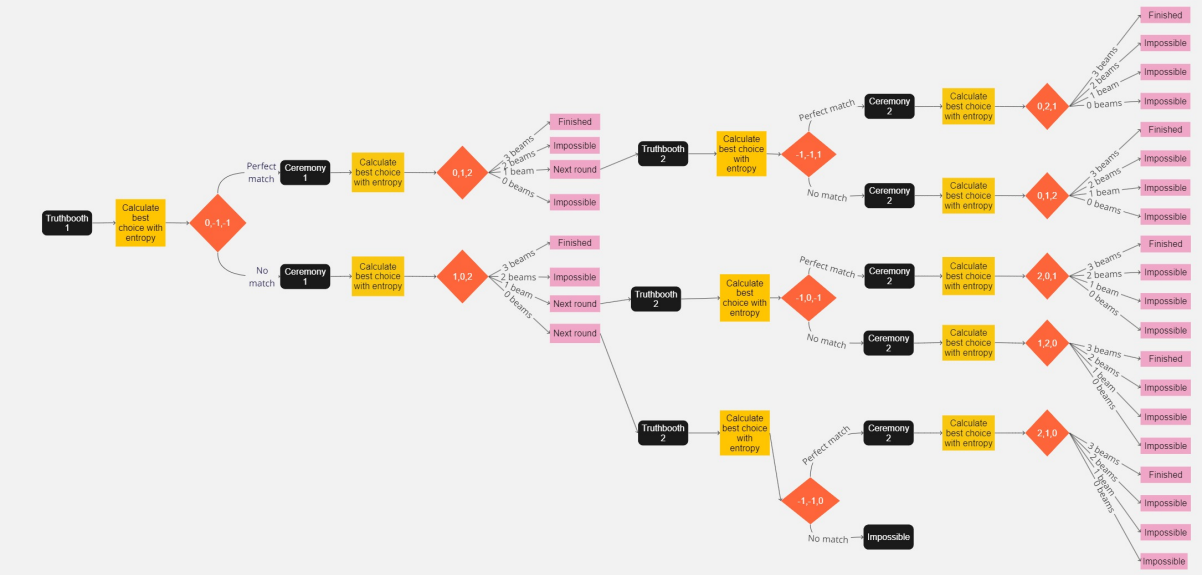

Figure 2.2: A game tree for 3 persons using entropy made in Miro

sentence to loop through x in pm and through y in pm while summing how many of x are the same as y. The sum_all is a naive sum function to utilize the parallel fast computing of numba and will do the exact same as np.sum($x == y$).

The second database is a matrix of all possible truth booths versus all possible matchups. All possible truth booths are again indicating the possible options and the matchups are all the possible solutions. This matrix consists of 1's and 0's depending on whether the match in the truth booth is present in that solution. This is also computed by sum_all_jit, however y loops over the truth booths instead of the matchups.

Note that for the first truth booth there are still 3 628 800 possible matchups for the greatest case of 10 couples. Calculating the best action is therefore expected to take a lot of time. However, since there is no information whatsoever for the first truth booth, every possible action will yield the same information (i.e. has the same entropy). To reduce computational complexity, the first truth booth is standardized to test man 0 with woman 0. The first steps are in this manner merely a notation; the first truth booth couple can be named as man 0 and woman 0 and the first matchup can be named "0,1,2,...,n-1" or "1,0,2,...,n-1" depending on the result of the first truth booth. By pre-calculating the outcomes of the first round (i.e. the first truth booth as well as the first match-up), we end up with 21 distinct possible situations: the first batch of options is that the first truth booth is a match and the first matchup is 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10 beams. The second batch of options is that the first truth booth is a no match and the first matchup is 0, 1, 2, 3, 4, 5, 6, 7, 8, 9 or 10 beams. This results in a split into 21 matrices for the two databases instead of one matrix that compares all the codes.

If n for the amount of couples is lower than 10, there are still 21 splits, although there will be at least one empty matrix. The ones for beams lit greater than n can not have any possible matchups, as there can not be more than n matches correct if there are not more than n couples. One of the other matrices will also never be filled; the matrix for n-1 beams will be empty since n-1 beams will never be lit for n couples. The reason for that is if only one couple would be incorrect, a switch between couples is not possible since all other couples are correct.

The splits are produced by comparing each solu-

Algorithm 2.3 Split for the first truth booth

```
firstsplit ⇐ new Array
firstquestion ⇐ [1, 0, ..0] with length n
for y in pm do
  if sum_all_jit(firstquestion is y) is 0 then
    append y to firstsplit[0]
  else
    append y to firstsplit[1]
  end if
end for
```

Algorithm 2.4 Split for the first matchup

```
splitarray ⇐ new Array
firstquestion ⇐ [1, 2, .., n]
secondquestion ⇐ [2, 1, 3, ..., n]
for y in firstsplit[0] do
  number ⇐sum_all_jit(secondquestion is y)
  append y to splitarray[number]
end for
for y in firstsplit[1] do
  number ⇐ sum_all_jit(firstquestion is y)
  append y to splitarray[10 + number]
end for
```

tion, with sum_all_jit again, to the first question [1,0,...0] with length n as stated in Pseudocode 2.3 and comparing each solution to [1,2,..n] or [2,1,3,...,n] depending on the outcome of the first truth booth as seen in Pseudocode 2.4. However, since numba does not work well with for loops for filling arrays, the second split is not actually produced by a for loop but written out. Further, there is technically the possibility to reach splitarray[10] by a perfect match in the first truth booth and 0 beams in the matchup ceremony and as well by a no match in the first truth booth and 10 beams in the matchup ceremony. Due to the strategy of the program the first possibility will never be reached anyway as the perfect match will be applied to have at least one beam and the array will only contain the solution of the second possibility.

The databases are saved in a npy file for deployment in the algorithm in the "Are you the one?" program.

2.3.3 Approach Likelihood: The gameplay

The greatest case of a 10 couple game has 3 628 800 different game plays, as there are that many different solutions and the program always decides the same decision with the same remaining options. As the game selects a random solution, those exact game plays cannot be simulated to be played exactly once. Therefore the game is implemented to run 1000 times for each amount of couples, as sample of those game plays, in Python. The game exists of 4 files; one *main* that loops through the games, one *game* that pulls the strings in the gameplay, one *datastore* that updates the databases and one *shannon* that makes the decision.

The *main* imports the game and keeps track of the amount of times it has won a game and how many rounds were needed in those games. The input is the amount of games *amt* and the amount of couples n. It loops *amt* times through the initiation of a Game(n), and if the game is won it adds the rounds of this game to the total rounds and 1 to the amount of wins.

Each *game* consists of the following characteristics:

- solution
- amount of couples
- part of a round
- amount of rounds
- Boolean solved
- truth booth database
- matchup database

The *game* consists of two functions: playing the game and playing one round. The game playing starts of with making a random permutation for the solution (np.random.permutation). The game switches between the truth booth round and the matchup round, which it keeps track of by the variable of the part of the round. It stops when in the matchup round the solution is found, which is checked by the Boolean solved, or the amount of rounds ran is the same as the amount of couples

(amount of rounds). It adds another round if lost to indicate that it had lost the game. The truth booth database and the matchup database start off with the matrix that corresponds with the gameplay of the first round from the npy files and is initiated after the first round.
Both rounds consists of the same steps. First the best option is calculated through entropy, except for the first round where the decision already is taken as it is always the same. Then this option is compared with the solution. From this an amount of beams or a 1 or 0 (perfect match or no match) emerges. The matrices are cleaned up through a function of the datastore file and the program switches to the other round.

The *datastore* file has 4 characteristics: an array of possible matchups, an array of possible truth booths and the two databases. In the initialization after the first round, only one of the 21 matrices that matched the outcome of the first round is loaded for the databases. It has a function to clean up the databases after the truth booth or matchup round. The input of the function are the part of the round (truth booth or match up), the decision that was taken and the amount of beams or either 1 or 0 for the truth booth. It checks which rows and columns are still viable through np.where and then keeps those rows and columns through np.take. The array for possible matchups is updated as well in this step.

The decision making file *shannon* imports the blockentropy mentioned earlier and has only one function. This function has as input which part of the round it is and the data of datastore. It loops through the database of matchups or truth booths depending on which part of round it is. It calculates for each row the entropy. If it is greater than the last highest entropy, it retains the entropy and which row it is. It reports the row of the biggest entropy as decision to be taken.

When all the games are played, the percentage of how many games are won (divided by the total amount of games) and the average of rounds needed (divided by the total amount of games won) are reported in the *main*. A list of the amount of rounds needed each time is printed as well.

3 Results

As the two programs, calculating the Maximum amount of rounds and calculating the Likelihood of amount of rounds, have two different approaches, they each generate different results. Approach M has ran only one time and has as a result a maximum amount of rounds. Approach L has ran 1000 times and has as a result a statistical outcome. These results are described in the next two subsections.

3.1 Rounds for approach Maximum

As the game tree in approach M develops all possible game plays, it only returns one number: the depth of the game tree. This depth is the maximum amount of truth booths and matchup ceremonies. For example, the solution for 6 couples is always at least found at the point when the contestants are deciding for the 6th truth booth (and after 5 ceremonies), which is a depth of 11. In spite of having the solution at the 6th truth booth, the game can not end by a truth booth, but only by a ceremony. The depth can be interesting to look at for how the game tree grows by adding another couple, but is not essential for the outcome of the game. The rounds, which are essential, can be found by dividing the depth by 2 and rounding up. Table 3.1 summarizes the results of the game tree.

Table 3.1: Rounds and depth of the game tree in approach Maximum to solve "Are you the one" for 1 to 8 couples. The complete game tree for 3 couples is shown in Figure 2.1.

Couples	Depth	Rounds
1	1	1
2	2	1
3	4	2
4	6	3
5	8	4
6	11	6
7	14	7
8	17	9

Approach M ran into problems with execution time for 9 and 10 couples. However, for 8 couples it is giving a maximum of 9 rounds, so it is already clear that for 8 couples and higher it cannot be solved in n rounds with n the amount of couples.

3.2 Rounds for approach Likelihood

The program for the Likelihood exists of two parts. The first part was the preparation for the game, the *datamaking*, and the second part the game itself. Due to memory problems, the *datamaking* program would stop running when it would make the first matrix for 9 couples as it had too many data. So it could only run up to 8 couples. Because of this, the game itself could also only run up to 8 couples. The results of the game are pictured in Figure 3.1.

The median of the rounds needed to get to a solution is two rounds less than the amount of couples for 6 to 8 couples. For 4 and 5 couples it is one round less. The dispersion of those rounds gets wider with each added couple. The rounds needed for 4 couples is for example between 1 and 3 rounds and the rounds needed for 8 couples is between 2 and 8 rounds. The maximum amount of rounds in the 1000 times running of the game is 8 for 8 couples. So it is possible that the maximum amount of rounds being limited by the amount of couples also holds for 9 and 10 couples, but it can not be stated with 100 % certainty.

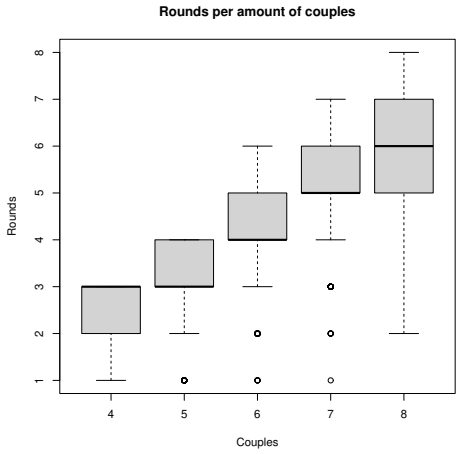

Figure 3.1: Rounds needed to solve "Are you the one". Each box summarizes 1000 times of running the game for 4, 5, 6, 7 and 8 couples

The table maxima of Table 3.1 matches with the statistics of the game. Each maximum is the same or higher. As the game does not explore every option, it is probably the case that it did not run into the solution that will cause the higher number of rounds.

4 Discussion and conclusion

To discover the amount of rounds of matchup ceremonies and truth booths needed to uncover the correct match-up of n couples, two different programs tested "Are you the one?" for 3 to 8 couples. A n-digit code represented the n couples each time and the choices in each round were decided through highest entropy. Approach Maximum explored a game tree to find the maximum of rounds needed and the approach Likelihood played the game 1000 times for a random solution to find out the statistical distribution of the needed rounds.

The number of rounds for both approaches were found up to 8 couples. In approach M the maximum number of rounds for 8 couples was 9 rounds. The maximum number of rounds for 8 couples by testing random solutions for 1000 runs in approach L was 8 rounds. However, some games for 8 couples in approach L could be solved in 2 rounds and the median was 6 rounds.

For 9 and 10 couples applying entropy as only method is not desirable in terms of speed and memory. Only a predicted number of rounds could be indicated. The prediction for approach M is that it cannot always be solved within n rounds for 9 and 10 couples. This is in contrast to the expectation of approach L with 10 couples; there is a great chance of solving it within 10 rounds, as most of the solutions probably can be found within the 10 rounds.

Although the maximum number of rounds are found for 8 couples and indicated for 9 and 10 couples, only for up to 7 couples the maximum number of rounds is at most the same number as the number of couples as seen in approach M (Table 3.1). However, it is most likely that if a random solution is set, that for 8, 9 or 10 couples the solution is found within 8, 9 or 10 steps (Figure 3.1 and derived from this figure).

To summarize all, it is most likely to uncover the the correct match-up of n couples using at most n rounds of matchup ceremonies and truth booths without external information with n up to 10 couples. In addition to this is it always possible to uncover this up to 7 couples. Of course, in the game show the players receive external information in the form of shared experience, which should improve their chances even more.

4.1 Future research

As "Are you the one?" itself is a variation of a game, it brought up a few questions during researching. Since the TV show had a few variations in the series itself, there are a few obvious other options to look at. The most intriguing questions that came up while discussing the game were "What can we do with the outcome?" and "What other variations of the game could be looked at?".

4.1.1 Find a strategy in data

So what can one do with the outcome? As one can see there is not a clear path to remember the game tree and use that strategy to play "Are you the one?". However, each path chosen by calculating entropy is one of multiple paths that can be chosen. The entropy of multiple paths can be the exact same, and sometimes the equally highest, score. Which path is chosen depends on the manner of programming. How the paths elapse can be influenced in three ways. Firstly, how the data of possible solutions and truth booths is ordered. Secondly, how the program loops through the data. Thirdly, which of the highest scores is chosen, this is often the first or last option in a loop.

If all those paths are explored, there might be an easy path and therefore strategy that can be remembered. As this would be another research question, the program would have to be expanded to be looking through all the possible paths instead of one of the evenly good ones. Since the program was already too slow for only determining the best outcome, the option to research a strategy that might work would cost even more time and memory. A solution can be found in finding a strategy for the lower numbers. This could be adjusted and applied to higher numbers.

4.1.2 Other varieties in the game show

As the game show has multiple seasons, the producers introduced some varieties to manufacture some interesting consequences. They introduced an eleventh girl, to have one guy with two possible matches. If you would exclude the girl, the season would elude the exact same strategy as usual. The consequences, without taking in account the psychological backslash of drama of an extra potential love, would be none therefore.

In addition, they established in subsequent seasons a penalty for each time the matchup ceremony had 0 beams, excluding the beams for already confirmed perfect matches. This has more effect, as the strategy would differ if you have to ensure the non-possibility of 0 beams. So this is a variation of the game show that can be examined in future research.

Another season featured 20 bisexual persons. This would relinquish the games similarity to Mastermind. As every contestant could be coupled up to every other contestant, the amount of possibilities would be a combination instead of a permutation. The function to calculate this would be:

$$\frac{20!}{2^{10} \cdot 10!} = 654\,729\,075.$$

This would mean that there are $\frac{654\,729\,075}{3\,628\,800} \approx 180$ times more possible solutions than with the original 10 pairs and probably a more complex strategy.

So, although this research already shows the possibilities in the traditional game show, strategies and variations of this show still can be investigated in the future.

References

L. Berghman, D. Goossens, and R. Leus. Efficient solutions for Mastermind using genetic algorithms. *Computers & Operations Research*, 36(6):1880–1885, 2009. ISSN 0305-0548. doi:10.1016/j.cor.2008.06.004.

D. E. Knuth. The computer as master mind. *Journal of Recreational Mathematics*, 9:1–6, 1977.

J. J. Merelo-Guervós, P. Castillo, A. M. M. García, and A. I. Esparcia-Alcázar. Improving evolutionary solutions to the game of mastermind using an entropy-based scoring method. In *Proceedings of the 15th Annual Conference on Genetic and Evolutionary Computation*, GECCO '13, page 829–836, New York, NY, USA, 2013. Association for Computing Machinery. ISBN 9781450319638. doi:10.1145/2463372.2463473. URL `https://doi.org/10.1145/2463372.2463473`.

E. Neuwirth. Some strategies for mastermind. *Zeitschrift für Operations Research*, 26:B257–B278, 1982.

D. Partynski. Cluster-based particle swarm algorithm for solving the mastermind problem. In *Proceedings of the International MultiConference of Engineers and Computer Scientists*, volume I, Hong Kong, March 2014.

L. Serano. *Shannon Entropy, Information Gain, and Picking Balls from Buckets*. 2017. `https://medium.com/udacity/shannon-entropy\-information-gain-and-picking-balls-from\-buckets-5810d35d54b4`, [Online; accessed 5-June-2023].

H. F. Ugurdag, Y. Sahin, O. Baskirt, S. Dedeoglu, S. Gören, and Y. S. Kocak. Population-based FPGA solution to mastermind game. *First NASA/ESA Conference on Adaptive Hardware and Systems (AHS'06)*, pages 237–246, 2006.