Master Minds: The Report

Xinhao Su
xs2413

David Xu
dx2199

December 21, 2021

1 Problem Overview

Master Mind is a classic codebreaking game first played in 1970. Gameplay goes as follows:

1. The "codemaker" picks four pegs and places them in order. This is the "solution code". Each peg can be one of six different colors and colors can be repeated between pegs.
2. The "codebreaker" then has 8 turns to guess this code. On each turn:
   (a) The codebreaker makes a guess of the code (four pegs in order, each having one of the six colors).
   (b) The codemaker responds with feedback using some number of black and white pegs. Each black peg means a peg of the guess is both the right color and in the right position. Each white peg means a peg of the guess is the right color but in the wrong position.
   (c) Using this information, the codebreaker can formulate his/her next guess.
3. The game ends after the code has been guessed (the codebreaker wins) or eight incorrect guesses have been made (the codemaker wins), whichever occurs first.

A photo of a physical game board is shown below, albeit with slightly different colors.

2 Formal Definition

This game can be formalized as The Mastermind Problem. Given a set of guesses and their corresponding feedback, can we determine what solutions code(s) would produce such behavior? This is effectively a multi-dimensional search problem which has been proven to be NP-Complete[1].

[1]Stuckman, J., & Zhang, G. Q. (2005). Mastermind is NP-complete. *arXiv preprint cs/0512049*.

3 Algorithm

Donald Knuth proposed the "Five-Guess Algorithm[2]", which was named as such because it will always determine the correct code using only at most five of the eight allowable guesses. The algorithm works as follows:

S.1 Generate all possible codes. Call this S, representing the universe of possible codes. Here, $|S| = 6^4 = 1296$.

S.2 Create a set of possible solutions P. At the start, $P = S$.

S.3 Choose an initial guess g_0. This can be hard-coded or randomly selected from P, as no information about the solution code has been obtained yet.

S.4 Play guess g_0 and obtain a response r from the codemaker.

S.5 Filter P to remove all codes which could not possibly be the solution based on this response.

   (a) For a code to possibly be the actual solution, its response to g_0 must also be exactly r.

S.6 Select the best next guess g_1 from S using a minmax algorithm: minimize the maximum possible number of remaining codes in P after this guess.

   (a) The maximum possible number of remaining codes in P after a guess g_2 can be calculated by determining the response of each code in P to g_2. The response which appears with the greatest frequency is the worst case (resulting in the largest P after filtering.

S.7 Repeat from **S.4** with guess g_1. Repeat until the solution is found ($|P| = 1$).

4 Extension

It is desirable to increase the computational difficulty of the problem so as to produce more reliable timing results. A program running over a longer amount of time will allow for recorded times to be more robust against random noise. As such, we increase the search space of the algorithm by extending the game to use c colors and h holes (previously, $c = 6$ and $h = 4$). As such, the universe of codes expands: $|S| = c^h$. For performance analyses in the remainder of this report, $c = 10$ and $h = 4$ are selected, resulting in $|S| = 10,000$.

5 Sequential Implementation

5.1 Datatypes

5.2 TODO – other sections

5.3 Performance

6 Parallelization

TODO: what we tried for parallelization, results, etc.

6.1 parMap

TODO: parMap each code comparison (too many sparks); then parMap each candidate code

[2] Knuth, D. E. (1976). The computer as master mind. *Journal of Recreational Mathematics, 9*(1), 1-6.

6.2 Chunks

7 Conclusion

8 References

9 Appendix: Code