The game of MOO

Andy Pepperdine

There is an old game called Moo, which was later commercialised under the name of Mastermind[2]. In the original form that I knew, the first player chooses four distinct digits in a row (the target). The order of the values in the row is relevant. The second player then proceeds to make a guess by suggesting what the four values are, and what the order is. The first player then says how many values are exactly right and in the right position (bulls), and how many values are correct, but in the wrong position (cows). So if the target is 4387, and the guess is 4871, then the reply would be 1 bull (the 4), and 2 cows (the 7 and 8).

Guesses continue until the second player guesses the target exactly right, scoring the number of guesses made. Players then alternate in choosing a target, either for a fixed number of games, adding up the total number of guesses, the lower number winning; or until after an even number of games, the difference in scores is greater than a previously agreed amount.

Questions

Is there an algorithm that can identify efficiently what guesses should be made?

What is the best strategy, in the sense that it will give the lowest average number of guesses for a game?

What is the best strategy in the sense that it gives the lowest number of guesses for the worst case?

This paper examines a few possible strategies that are easily computable, and reports a computer analysis of each.

Comment

A method of computing the optimal strategy can be found at [4] (in Japanese), and this takes now a “reasonable” amount of computing time. When I first heard of the game, it would have been an unreasonable amount of time without access to extensive computer facilities.

A paper in English giving an optimal strategy requiring an average number of guesses of 4.340, with a maximum of 6 is given by Koyama and Lai in 1993 [6]. A slight modification, also reported in [6], reduces the maximum number of guesses to 5, but increases the average to 4.341.

More information on the game can be found at Mathworld [7].

Terminology

At a particular stage in the game, a number of guesses have been made, and the possible values for the target have been reduced according to those which would give the answers found so far.

Let A be the set of all possible targets; and T be the set of possible targets at a given point in the game.

Let g be a guess to be made. It will divide T into 14 subsets such that all members of each subset give the same response when challenged by g, and no two elements from different subsets give the same response to g. The 14 answers are: BBBB, BBB, BBCC, BBC, BB, BCCC, BCC, BC, B, CCCC, CCC, CC, C, 0. Some subsets may be empty, and these will be ignored.

Let the number of non-empty subsets be n, and the subsets be $\{t_i\}$, $1 <= i <= n$. The number of elements in $\{t_i\}$ is $|t_i|$.

Let's suppose a strategy is used that does no look-ahead. We can represent that by a value function V(g,T), and without loss of generality, we can look for the set G = {g : V(g,T) is a minimum}. Then choose one element of G as the next guess to make.

If we restrict g ∈ T, then the results may in fact be less effective than considering g ∈ A. The results given here assume g ∈ A.

Exhaustive analysis

Because G contains more than one element usually, it does in fact matter which one we choose, although in practice it appears that they vary little in their effects. The average number of guesses for a game seems to vary by about 0.04% depending on the algorithm used to choose the element of G.

If g is chosen from T, then one might expect that the number of guesses would be reduced, but in fact that is often not the case. Also, it is not always the case that it reduces the maximum number of guesses to make. The exceptions are noted in the table below.

All guesses at the beginning of the game are equivalent, so we can choose any. Thereafter, we select according to minimising V(g,T) and can do a complete search recording the maximum number of guesses and the average when applied to all targets. The results are tabulated below.

Description	Function, V, to minimise	Ave.	Max.	Notes
Choose any g ∈ T	$1-\tau(1)$	5.445	8	1
Maximum size of a subset	$\max\lvert t_i\rvert-\tau(1)$	5.380	7	2, 3
Exponential asymptote	$\{\sum_{1\leqslant i\leqslant n}\lvert t_i\rvert\cdot(1-e^{-\lvert t_i\rvert})\}-\tau(1-e^{-1})$	5.363	8	
Maximise sum of reciprocals	$\frac{1}{\sum_{1\leqslant i\leqslant n}\lvert t_i\rvert^{-1}}$	5.352	8	4
Maximise geometric mean	$\frac{-\sum_{1\leqslant i\leqslant n}\ln(\lvert t_i\rvert)}{n}-2\cdot\ln(n+\tau(1))$	5.328	7	2
Minimise the square of the variance	$\frac{\sum_{1\leqslant i\leqslant n}\lvert t_i\rvert^2}{n}-\tau(1)$	5.265	8	
Logarithmic integral	$\{\sum_{1\leqslant i\leqslant n}\lvert t_i\rvert\cdot\mathrm{li}(1+\lvert t_i\rvert)\}-\tau(2\cdot\mathrm{li}(3))$	5.247	8	5
Landy function	$\sum_{1\leqslant i\leqslant n}\lvert t_i\rvert\cdot\mathrm{L}(\lvert t_i\rvert)$	5.246	8	6
Square root	$\{\sum_{1\leqslant i\leqslant n}\lvert t_i\rvert\cdot\sqrt{\lvert t_i\rvert}\}-\tau(1)$	5.244	8	7
Natural log (1+t)	$\{\sum_{1\leqslant i\leqslant n}\lvert t_i\rvert\cdot\ln(1+\lvert t_i\rvert)\}-\tau(2\ln 2)$	5.242	8	3
Information content	$\{\sum_{1\leqslant i\leqslant n}\lvert t_i\rvert\cdot\ln(\lvert t_i\rvert)\}-\tau(2\ln 2)$	5.239	8	8

Notes

1. The function τ(x) ⟶ x if g ∈ T, 0 otherwise. This is a useful function for adjusting the result for the case when the guess made hits the answer rather than having to make another guess.
2. If we take g ∈ T, then the average is somewhat less, but the maximum number of guesses increases to 8.
3. This is one of three functions suggested by Mr B. Landy [3].
4. This function attempts to spread the results among as many subsets as possible.
5. The logarithmic integral is defined as $\mathrm{li}(x)=\int_0^x\frac{dt}{\ln(t)}$, (x > 1)

6. The Landy function is defined as $L(x)=z, z^z=x$, and was introduced in [3] by Mr B. Landy.
7. This is a surprisingly good result. There is no obvious reason why the square root function should split the sets up appropriately.
8. Dr J. Larmouth suggested this based on information theory [3].

Acknowledgements

When the first edition of this paper was written, I was not aware of anyone having a complete analysis. Since then [4] has been brought to my attention by private correspondence from John Francis on 2009-12-10. He further claims that he had also much earlier found a complete strategy, but only now is in the process of preparing it for publication. I wish to thank him also for correcting a few matters of fact. You can find his analysis at [5].

References

[1] Creative Commons License, *http://creativecommons.org/licenses/by/2.5/*

[2] http://www.pressmantoy.com/

[3] $\aleph_0$: Computer Recreations, *Software Practice and Experience*, Vol. 1, No. 2, pp. 201-204 (1971).

[4] Tetsuro Tanaka, An optimal MOO strategy, found at http://www.tanaka.ecc.u-tokyo.ac.jp/~ktanaka/papers/gpw96.pdf (in Japanese)

[5] John Francis, *Strategies for playing MOO, or "Bulls and Cows"*, 2010, http://www.jfwaf.com/Bulls%20and%20Cows.pdf

[6] Kenji Koyama, Tony Lai, An Optimal Mastermind Strategy, *Journal of recreational Mathematics*, vol 25/4, pp 251 – 255 (1993)

[7] MathWorld, http://mathworld.wolfram.com/Mastermind.html

2010-01-29