Finding an evolutionary solution to the game of MasterMind with good scaling behavior

Juan J. Merelo, Antonio M. Mora[1], Carlos Cotta, Antonio J. Fernández-Leiva[2]

[1] Dept. Computer Architecture and Technology + CITIC
University of Granada
`(jmerelo|amorag)@geneura.ugr.es`
[2] Dept. of Computer Sciences and Languages
University of Málaga
`(ccottap|afdez)@lcc.uma.es`

Keywords: MasterMind, oracle games, puzzles, evolutionary algorithms, parameter optimization

Abstract. The main research issue in the game of MasterMind is to find a method that minimizes the number of turns needed to find the solution. But another venue is to find a method that scale well when the size of the search space is increased. In this paper we will present a method that uses evolutionary algorithms to find fast solutions to the game of MasterMind that scale better with problem size than previously described methods; this is obtained by just fixing one parameter. We prove its scalability by testing it over a wide range of MasterMind problem sizes.

1 Introduction and state of the art

MasterMind [2] is a puzzle in which one player A hides a combination of κ symbols and length ℓ, while the other player B tries to discover it by playing combinations using the same alphabet and length. The answers from player A to every combination include the number of symbols in the combination that are in the correct position and the number of colors that have been guessed correctly. Player B then xplays a new combination, until the hidden one is found. The objective of the game is to play repeatedly minimizing the number of turns needed to find the solution.

Most MasterMind algorithms so far [3, 4] use the concept of *eligible*, *possible* or *consistent* combinations: those that, according to responses by player A, could still be the hidden combination or, in other words, those that match the played combinations as indicated by the answer. Exhaustive methods [2, 5] would eliminate all non-consistent solutions and play a consistent one, while non-exhaustive methods would sample the set of consistent solutions and play one of them. Those solutions are guaranteed to reduce the search space at least by one, but obviously different combinations have a different reduction capability. This *capability* is reflected by a score. However, scores are heuristic; there is no rigorous way of scoring combinations. To compute these scores, every combination is compared in turn with the rest of the combinations in the set; the number of combinations that get every response (there is a limited amount of possible responses) is noted. Eventually this results in a series of *partitions* in which the set of consistent combinations is divided by its *distance* (in terms of common positions and colors) to every other. From this a set of combinations with the best score is obtained; one of the combinations of this set is chosen deterministically (using lexicographical order, for instance) or randomly. In this paper we use *most parts*, proposed in [6], which takes into account only the number of non-zero partitions.

Currently, the state of the art was established by Berghman et al. in [3]. They obtained a system that is able to find the solution in an average number of moves that is, for all sizes tested, better than previously published. The number of evaluations was not published, but time was. In both cases, their solutions were quite good. However, there were many parameters that had to be set for each size and no systematic method to set them, starting with the first guess and the size of the consistent set, as well as population size and other evolutionary algorithm parameters. In this paper we will try to adapt our previously published Evo method by reducing the number of parameters without compromising too much on algorithm performance, based on the fact that even as you can find a good solution using only a sample of the consistent set size as proved in [3, 4], different set sizes do have an influence on the outcome. When you reduce the size to the minimum it is bound to have an influence on the result, in terms of turns needed to win and number of evaluations needed to do it. The effect of the reduction of this sample size will decrease the probability of finding, and thus playing, the hidden combination, and also the probability of finding the combination that maximally reduces the search space size when played. However, in this paper we will prove that good solutions can be found by using a small sample size and, what is more, a common set size across all MasterMind problem sizes.

In the next section we will present the experiments carried out and its results for sizes from $\ell = 4, \kappa = 8$ to $\ell = 7, \kappa = 10$.

2 An evolutionary method for playing MasterMind

This paper uses the method called, simply, *Evo* [7–10], which has been released as open source code at CPAN (`http://search.cpan.org/dist/Algorithm-MasterMind/`) and is an evolutionary algorithm that has been optimized for speed and to obtain the solution in the minimal number of evaluations possible. An evolutionary algorithm [11] is a Nature-inspired search and optimization method that, modeling natural evolution and its molecular base, uses a (codified) population of solutions to find the optimal one. Candidate solutions are scored according to its closeness to the optimal solution (called *fitness*) and the whole population evolved by discarding solutions with the lowest fitness and making those with the highest fitness reproduce via combination (crossover) and random change (mutation).

Evo, which is explained extensively in [10] searches consistent combinations until a prefixed amount of them has been found. It uses *Most Parts* score to assess consistent combinations, and the *distance to consistency* for non-consistent ones, so that the fitness directs search towards finding consistent solutions with better score. The algorithm continues until a pre-fixed number of consistent solutions have been found or until this number does not vary for a number of generations (set to three throughout all experiments).

Evo incorporates a series of methods to decrease the amount of evaluations needed to find the solution, including *endgames* which makes the evolutionary algorithm revert to exhaustive search in the case the search space has been well characterized (for instance, when we know that the solution is a permutation of one of the combinations played or when we have discarded some colors, reverting to a problem of smaller size).

The solutions are quite promising, but the main problem is that the number of evaluations needed to find the solution increases rapidly with problem size (fortunately, not as fast as the problem size itself or this solution would not be convenient) and a new parameter is introduced: the optimal size of the set of consistent combinations, that is, the number of combinations that the algorithm tries to find out before it plays one.

What we do in this paper is testing an *one size fits all* approach by making the size of the consistent set unchanged for any problem size. This reduces the algorithm parameter set by one, but since this parameter set has, a priori, a big influence on result and there is no method to set it other than experimentation, it reduces greatly the amount of experiments needed to obtain a reasonable solution.

3 Experiments and results

The experiments presented in this paper extend those published previously, mainly by [10].

In this paper we will set this size to a minimal value common for all sizes: 10, that is why we will denominate the method tested Evo10. This value has been chosen to be small enough to be convenient, but not so small that the scoring methods are rendered meaningless. This will reduce the parameters needed by one, leaving only the population size to be set, once, of course, the rest of the evolutionary algorithm parameters have been fixed by experimentation; these parameters are set to crossover rate equal to 80%, and mutation and permutation rate equal to 10%; replacement rate is equal to 75% and tournament size equal to 7.

For every problem size, a fixed set of 5000 combinations were generated randomly. There is at most a single repetition in the smallest size, and no repetition in the rest. The sets can be downloaded from `http://goo.gl/6yu16`; these sets are the same that have been used in previous papers. A single game is played for every combination.

Table 1. Comparison among this approach (*Evo10*) and previous results published by the authors (Evo++) in [10] and Berghman et al. [3].

(a) Mean number of guesses and the standard error of the mean for $\ell = 4, 5$, the quantities in parentheses indicate population and consistent set size (in the case of the previous results).

	$\ell = 4$	$\ell = 5$	
	$\kappa = 8$	$\kappa = 8$	$\kappa = 9$
Berghman et al.		5.618	
Evo++	(400,30) 5.15 ± 0.87	(600,40) 5.62 ± 0.84	(800,80) 5.94 ± 0.87
Evo10	(200) 5.209 ± 0.91	(600) 5.652 ± 0.818	(800) 6.013 ± 0.875

(b) Mean number of guesses and the standard error of the mean for $\ell = 6, 7$, the quantities in parentheses indicate population and consistent set size (in the case of the previous results).

	$\ell = 6$		$\ell = 7$
	$\kappa = 9$	$\kappa = 10$	$\kappa = 10$
Berghman et al.	6.475		
Evo++	(1000,100) 6.479 ± 0.89		
Evo10	(800) 6.504 ± 0.871	(1000) 6.877 ± 0.013	(1500) 7.425 ± 0.013

(c) Mean number of evaluations and its standard deviation$\ell = 4, 5$.

	$\ell = 4$	$\ell = 5$	
	$\kappa = 8$	$\kappa = 8$	$\kappa = 9$
Evo++	6412 ± 3014	14911 ± 6120	25323 ± 9972
Evo10	2551 ± 1367	7981 ± 3511	8953 ± 3982

(d) Mean number of evaluations and its standard deviation$\ell = 6, 7$.

	$\ell = 6$		$\ell = 7$
	$\kappa = 9$	$\kappa = 10$	$\kappa = 10$
Evo++	46483 ± 17031		
Evo10	17562 ± 135367	21804 ± 67227	40205 ± 65485

The results for this fixed parameter setting are shown in Tables 1(a),1(b), 1(c) and 1(d).

The first of these tables, which represent the average number of moves needed to find the solution, shows results that are quite similar for boths methods. The average for Evo10 is consistently higher (more turns are needed to find the solution) but in half the cases the difference is not statistically significant using Wilcoxon paired test. There is a significant difference for the two smaller sizes ($\ell = 4, \kappa = 8$ and $\ell = 5, \kappa = 8$), but not for the larger sizes $\ell = 5, \kappa = 9$ and $\ell = 6, 7$. This is probably due to the fact that, with increasing search space size, the difference among 10 and other sample size, even if they are in different orders of magnitude, become negligible; the difference between 10% and 1% of the actual sample size is significant, but the difference 0.001% and 0.0001% is not.

However, the difference in the number of evaluations (shown in Tables 1(c) and 1(d)), that is, the total population evaluated to find the solution is quite significant, with Evo10 needing from a bit less than half to a third of the total evaluations for the larger size. This means that the time needed scales roughly in the same way, but it is even more interesting to note that it scales better for a fixed size than for the best consistent set size. Besides, in all cases the algorithm does not examine the full set of combinations, while previously the number of combinations evaluated, 6412, was almost 50% bigger than the search space size for that problem. The same argument can be applied to the comparison with Berghman's results (when they are available); Evo++ was able to find solutions which were quite similar to them, but Evo10 obtains an average number of turns that is slightly worse; since we don't have the complete set of results, and besides they have been made on a different set of combinations, it is not possible to compare, but at any rate it would be reasonable to think that this result is significant.

4 Discussion, conclusions and future work

This paper has shown that using a small and fixed consistent set size when playing MasterMind using evolutionary algorithms does not imply a deterioration of results, while cutting in half the number of evaluations needed to find them. This makes the configuration of the algorithm shown quite suitable for real-time games such as mobile apps or web games; the actual time varies from less than one second for the smallest configuration to a few seconds for the whole game in the biggest configuration shown; the time being roughly proportional to the number of evaluations, this is at least an improvement of that order; that is, the time is reduced by 2/3 for the $\kappa = 6, \ell = 9$ problem, attaining an average of 4.7 seconds, almost one fourth of the time it can take when we try to achieve the minimum number of turns, 18.6 seconds. This number is closer to the one published by [3] for this size, 1.284s, although without running it under the same conditions we cannot be sure. It is in the same ballpark, anyways. The time needed to find the solution has a strong component in the number of evaluations, but it also depends on the consistent set size, that is why the relation between the time needed (1/4) is smaller than the relation between number of evaluations (roughly 1/3, see Table 1(d)). This allows also to extend the range of feasible sizes, and yields a robust configuration that can be used throughout any MasterMind problem.

As future lines of work, we will try to reduce even more this size and try to check whether it offers good results for bigger sizes such as $\ell = 7, \kappa = 11$ or even $\ell = 8, \kappa = 12$. Several consistent set sizes will be systematically evaluated, looking mainly for a reduction in the number of evaluations, and time, needed. Eventually, what we are looking is for a method that is able to resolve problems with moderate size, but this will need to be tackled from different points of view: implementation, middle-level algorithms used, even the programming language we will be using. We might even have to abandon the paradigm of playing always consistent solutions to settle, sometimes, for non-consistent solutions for the sake of speed.

It is also clear than, when increasing the search space size, the size of the consistent set will become negligible with respect to the actual size of the consistent set. This could work both ways: first, by making the results independent of sample size (for this small size, at least) or by making the strategy of extracting a sample of a particular size indistinguishable from finding a single consistent combination and playing it. As we improve the computation speed, it would be interesting to take measurements to prove these hypotheses.

Acknowledgements

This work is supported by projects TIN2011-28627-C04-02 and TIN2011-28627-C04-01 and -02 (ANYSELF), awarded by the Spanish Ministry of Science and Innovation and P08-TIC-03903 and P10-TIC-6083 (DNEMESIS) awarded by the Andalusian Regional Government.

References

1. Knuth, D.E.: The computer as Master Mind. J. Recreational Mathematics **9**(1) (1976-77) 1–6
2. Berghman, L., Goossens, D., Leus, R.: Efficient solutions for Mastermind using genetic algorithms. Computers and Operations Research **36**(6) (2009) 1880–1885
3. Runarsson, T.P., Merelo, J.J.: Adapting heuristic Mastermind strategies to evolutionary algorithms. In: NICSO'10 Proceedings. Studies in Computational Intelligence, Springer-Verlag (2010) 255–267 Also available from ArXiV: `http://arxiv.org/abs/0912.2415v1`.
4. Guervós, J.J.M., Mora, A.M., Cotta, C., Runarsson, T.P.: An experimental study of exhaustive solutions for the mastermind puzzle. CoRR **abs/1207.1315** (2012)
5. Kooi, B.: Yet another Mastermind strategy. ICGA Journal **28**(1) (2005) 13–20
6. Cotta, C., Merelo Guervós, J., Mora García, A., Runarsson, T.: Entropy-driven evolutionary approaches to the Mastermind problem. In Schaefer, R., Cotta, C., Kolodziej, J., Rudolph, G., eds.: Parallel Problem Solving from Nature PPSN XI. Volume 6239 of Lecture Notes in Computer Science., Springer Berlin / Heidelberg (2010) 421–431
7. Merelo, J., Mora, A., Runarsson, T., Cotta, C.: Assessing efficiency of different evolutionary strategies playing mastermind. In: Computational Intelligence and Games (CIG), 2010 IEEE Symposium on. (August 2010) 38–45
8. Merelo-Guervós, J.J., Cotta, C., Mora, A.: Improving and Scaling Evolutionary Approaches to the MasterMind Problem. In Chio, C.D., Cagnoni, S., Cotta, C., Ebner, M., Ekárt, A., Esparcia-Alcázar, A., Guervós, J.J.M., Neri, F., Preuss, M., Richter, H., Togelius, J., Yannakakis, G.N., eds.: EvoApplications (1). Volume 6624 of Lecture Notes in Computer Science., Springer (2011) 103–112
9. Merelo-Guervós, J.J., Mora, A.M., Cotta, C.: Optimizing worst-case scenario in evolutionary solutions to the MasterMind puzzle. In: IEEE Congress on Evolutionary Computation, IEEE (2011) 2669–2676
10. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2003)