Mastermind Project Design

Mark Pallone Max Spector

November 7, 2011

1 Generic Solver

Alexandre Temporel and Tim Kovacs[1] have presented a hill climbing algorithm for Mastermind which uses and supposedly improves upon research done by J. J. Merelo and Luís Bento. While the algorithm is scalable and shown to be reasonably efficient on non-standard board sizes (the largest being 6 pegs and 8 colors), no indication of the general scalability of the algorithm is presented in the paper.

For this phase of the final project, we shall implement this algorithm with two goals in mind:

- to discover how well the hill climbing algorithm works on very large peg and color domains, and
- to attempt to modify the algorithm to work better in larger environments.

2 Scalability

As mentioned in the previous section, we will attempt to modify the algorithm to work in larger environments. The paper itself makes several suggestions to generally improve the algorithm:

- allowing more color duplication after the intial guesses,
- selecting vastly different second and third guesses if the inital guess receives a low score,
- selecting vastly different colors from the previous two guesses if those guesses receive low scores, and
- replacing random guess generation with a pool generated by some genetic algorithm.

While the first three of these appear relatively straightforward, the final suggestion leaves much room for creativity, as several different fitness functions may be used (perhaps even based on the bias of the code generator).

Additionally, we wish to make the algorithm more suitable for larger search spaces by looking for ways to cheapen heuristic and guess-generation functions.

3 Learning

We shall learn encoder's biases by applying statistical tests of different types to the training data in order to attempt to classify the biases. We choose to use specific statistics instead of more general methods such as feature extraction because the biases are fairly simple and formulaic in nature. We first take the positive training data and run each statistic. We obtain from the statistic a correlation value, i.e., how much we feel the data correlates with one of our bias types. We then run each statistic on the negative training data and measure how much we feel that the statistic negatively correlates with the negative data.

For example, our first statistic is color usage, i.e., how often is a color used by the picker. With this statistic we attempt to pick up biases similar to the first test. If we were to run this test on the first set of positive training data, we should

[1]Alexandre Temporel, Tim Kovacs, A heuristic hill climbing algorithm for mastermind. Proceedings of the 2003 UK Workshop on Computational Intelligence (UKCI-03). ISSN 0862925371, pp. 189.196. September 2003. PDF, 109 Kbytes.

find about 25% usage for W, B, V, and I, and 0% usage for R,G,O, and Y. Now the exact number for the first set does not matter, but the 0% for the second set does. We can see a very high probability that only the last 4 colors are used, and the first 4 are not. We then run the same statistic on the negative data, and see an opposite result, 25% for R, G, O, and Y and 0% for W, B, V, and I. Finally, we take the difference and see that since all of our predictions are very far away from the negative of our predictions, we can have a high confidence that they are correct.

Our next statistic would be for finding biases similar to bias 2. We take a count of what percentage of positive test cases have a single red, double red, triple red. etc, and the same for each color. We note, for example, that statistically in the exact bias of test 2, the color does not matter for duplication. However, it may matter for similar cases; thus we shall check if it does statistically or not. We then run the same statistic on the negative test, subtract, and see that again we would be highly positive in the positive tests, and highly negative in the negative tests. Note that if we were running this statistic on test1, we would see a lot of color usage for the last 4 colors, and none for the first 4; the duplication numbers should be almost evenly distributed, with a slight spike around doubles. Thus we could attempt to say that there is a double duplicate bias for the first test. But, running the negative test would show that there is also a double duplication–but just for other colors. However, since the difference would be small, we would give the second type of bias a low confidence value, whereas the first one would receive a much higher confidence value.

For the third type of bias we simply count the number of colors, which can be expanded to some kind of color variety function as well. This color variety function can be used for test4. However, since test4 is a much more complicated bias, we will not see a huge correlation to any 1 color count, but we may see that the color variety value would be low.

All of these statistics are then compared, the one with the highest confidence value is chosen, and it is used to modify the values for the heuristic function described in problem 1. Biases of style 1 for example would have a 0 value placed for colors that did not fit, thus they would be skipped. Alternatively they can be added to the taboo list.

We include as a final attempt for classification a large amount of general statistical tests in order to modify the heuristic function values from the method used in problem 1. If we are unable to classify the bias, we can at least probabilistically increase the slope of the hill in the hill climbing. While this is not an optimal solution, it should still show improvement and only be used only on occasions in which we cannot classify.