Machine Learning Using a Genetic Algorithm to Optimise a Draughts Program Board Evaluation Function

Kenneth J. Chisholm and Peter V.G. Bradbeer.
Department of Computer Studies,
Napier University,
Edinburgh, Scotland.
e-mail: {ken,pvgb}@dcs.napier.ac.uk

Abstract

This paper reviews the authors' recent work in using a Genetic Algorithm (GA) to optimise the board evaluation function of a game-playing program. The test-bed used for this study has been the game of draughts (checkers). A pool of draughts programs are played against each other in a round-robin (all-play-all) tournament to evaluate the fitness of each 'player' and a GA is used to preserve and improve the best performers. Some solutions to the problems of attempting to compare the *absolute* performance of possible solutions in this area which is mainly about *relative* abilities are presented. Comparisons with classical methods and results are also briefly discussed.

1 Introduction

Although the original work of Samuel[Samuel59] on machine-learning using the game of draughts is almost 40 years old, the more recent work of the *Chinook*[Schaeffer92] checkers program team at the University of Alberta in the 1990's inspired the authors to look at the problem of optimising the board-evaluation function using a Genetic Algorithm (GA). The *Chinook* team has now virtually 'solved' the draughts end-game by creating an enormous database of solved boards[Lake94], and the openings are relatively well understood. This leaves the mid-game as an area where pure processing power and prodigious storage is not sufficient, due to the size of the search space. Thus it was felt that this game-playing area was still worth investigating using GAs to see if such an approach could improve on, or at least match, Samuel's rather tailored technique of machine-learning which helped enable his program to play at county level.

2 Historical Background: Basic Game-Playing Algorithms

Most standard game-playing programs for two-person, zero-sum board games such as draughts and chess use a limited look-ahead tree[Shannon50a] with mini-max search[Levy91], usually with some form of tree-pruning such as alpha-beta cut-off[Knuth75] to reduce the number of moves considered.

A board evaluation function is used for the terminal boards at the horizon (or leaf nodes) of the search tree. The principle of hot-pursuit[Turing53] is usually used so that the search-tree is locally extended to ensure that the boards which are evaluated by the board-evaluation function are relatively stable boards and are not so badly effected by the horizon effect[Berliner73]. This principle essentially causes all pending takes, for example, to be completed before a board is considered for the purposes of static board evaluation. This is particularly important in draughts since the rules of the game dictate that take-moves *must* be carried out by a player if any are available on a board. For more details of these issues see the papers of Turing[Turing53] and Shannon [Shannon50b] which discuss algorithms for playing computer chess, which all equally apply to draughts.

The static board evaluation function is essentially a weighted-sum of features score based on the various properties of the board. The board features considered when evaluating a terminal board are the usual properties thought important by human players such as: number of pieces, mobility count, centre-control, advancement of pieces, etc.

Thus the evaluated score for a board may be viewed as a simple linear polynomial, usually represented as follows:

$$\text{Board Score} = w_1 * f_1 + w_2 * f_2 + \ldots\ldots\ldots\ldots + w_n * f_n \qquad (1)$$

The features (f_1 .. f_n) used in such a board evaluation function are usually based on the human strategic knowledge of the game due to decades (or even centuries) of analysis[Belasco73, Fortman82]. However, the relative weights (w_1 ... w_n) assigned to these features can still be fruitfully analysed using optimising techniques such as hill-climbing. This paper reports how a GA can also be used to optimise and customise these weightings. Samuel in his first paper presented a very impressive method for (what he called) generalised learning using the technique of hill-climbing. This was certainly a considerable tour-de-force for its time. Samuel also reported how various board features could be viewed as being connected in some way (such as centre control and mobility, for example) and treated as a single feature for the purposes of the analysis. For practical reasons, the number of these so-called binary-connected terms which Samuel considered had to be limited and somewhat hard-coded into the program.

In his second paper, Samuel [Samuel67] presented a novel method of further grouping features together using what he called “signature tables”. This allowed the connectivity between features to be further extended to include tertiary and even hierarchical groupings. It is believed by the authors that the nature of the GA approach to this search space enables the connectivity of the various features to be captured and produced in a fairly direct manner.

3 A Genetic Algorithm for Optimising the Board Evaluation Function

The approach taken here is to try and evolve a set of weights for the polynomial in equation (1) with the weights themselves forming the genetic material processed by the GA[Holland75][Goldberg89], as a direct representation. To this end a draughts playing program is required to assess the quality of the candidate solution. A program called DRAFT5 was pressed into service. DRAFT5 is a descendant of a draughts playing program which was written by one of the authors almost 20 years ago as an undergraduate AI project[Chisholm76], to investigate various methods of achieving better rote-learning techniques, such as partial-board matching during end-games. This program has been extended, tuned, translated and ported over the years. DRAFT5 has been the main engine of the research reported here. DRAFT5 is written in ANSI C and currently runs on UNIX systems and PCs under MS-DOS. It has a graphical front-end and can suggest moves for the human opponent.

As with most draughts and chess programs DRAFT5 was written so that it was able to play against itself by alternately searching and moving for black and then for white. This capability was necessary for the purposes of the rote-learning experiments mentioned above. This ability of DRAFT5 would enable a pool of individuals with different weights to play against each other in some sort of draughts tournament such as the round-robin format. In practice by simply having a collection of differing weights in a two-dimensional array it is possible to effectively have a pool of ‘players’ with different strategies, due to each individual using its own set of weights in its board-evaluation function. (See figure 1 for a sample set of individuals. Note that the range figure underneath the feature descriptor indicates the maximum possible value permitted for that weight.)

The fitness function used in this GA is simply the number of wins achieved in a draughts round-robin (i.e. all-play-all) tournament by that individual in the pool. This is a measure of the *relative* fitness of the individuals in the pool. This clearly introduces an element of direct competition between the members of the pool. This can be seen as a similar model to that used by Rosin and Belew[Rosin95] except that one gene pool is used here, rather than the two distinct co-evolving populations (*hosts* and *parasites*) which they use in their GA model.

```
----------------------------------------------------------------Generation: 6
Fitness Piece King  Mob  KMob  Back  Cent  KCent Adv1  DBLSq  Near Exch  MOVE
Range:   1200 2000  140   140   140   140   140   140   660     5  140   240
-----------------------------------------------------------------------------
   23  1081  1917    17    25    22     1   110    85    85     4  128     7
   22   416  1564    85    79   131    49   118    65   349     1  104    70
   21   896  1564    17    25    74    49   118   138   230     1  104    70
   20   734  1068    86    91    74    12    69    65   365     5  136   163
   19   786   630    23   137    57    70    35    17    75     1  118   171
   18   896  1564     7    91    74    49   118   138   230     1  104   150
   18   416  1621    23    91    26    49    47    65    85     1  104   150
   ..   ...  ....   ...   ...   ...   ...   ...   ...   ...   ...   ...   ...
=============================================================================
```

Figure 1 - A sample pool of board feature weights

It should be noted that a round-robin(RR) tournament is used, as opposed to a simple knockout tournament for example, so that sufficiently accurate evidence is gathered regarding the ability of an individual program in the pool. This unfortunately causes a lot more games to be played per generation, thus slowing down each experiment, but this RR approach is thought to be absolutely necessary due to the high probability of draughts games ending in a draw. This is a very common feature in draughts tournaments, even with human players, and particularly with automatic game-playing programs such as DRAFT5. For example, in the Tinsley-*Chinook* match for the World Draughts Championship in 1993[Schaeffer93] which was played over 40 games, there were 33 draws, 6 wins for Dr. Tinsley and 2 wins for *Chinook*, leaving one game not played at the end of the match. However, it should be noted that draws are not quite so common in games involving opponents who are not of this exceptional calibre. The reward of half a point for a draw was discounted as it was believed that this would reduce the effectiveness of searching for aggressive end-game strategies.

In this first set of experiments a simple assortative selection and cross-over technique was used and elitism was employed to preserve the best “player” from each generation. The basic GA used is described in figure 2.

4 Results and Analysis of Initial Experiments

A first batch of experiments were conducted using the GA described in figure 2. Each experiment was carried out 10 times in an attempt to minimise the ‘noise’ in the results from DRAFT5.

```
/* Basic Draughts tournament GA - version 1 */
Max_Number_Of_Generations = 50
Pool_Size = 30
Mutation_Rate = 10

 /* Initialise the pool of  weights with random numbers */
 for i = 1 to Pool_Size do
   for j = 1 to Chromosome_length do
       Pool[i, j] = random(allowed range)

/* Carry our generations of RR draughts tournaments with GA */
for  g = 1 to Max_Number_of_ Generations do
{   /* Evaluate fitness of Pool of draughts players (using RR matches) */
    for i = 1 to Pool_Size do {
       for j = 1 to Pool_Size do {
           if (i # j) then  {
                  Copy Pool[i] weights to ‘Player A’ evaluation function
                  Copy Pool[j] weights to ‘Player B’ evaluation function
                  result = draughts ( Player A  vs. Player B)
                  if result = 1 then Pool[i].fitness = Pool[i].fitness + 1
                  if result = 2 then Pool[j].fitness = Pool[j].fitness + 1
             }
        }
    }
    Sort Pool of Players based on the Fitness from RR Draughts Tournament
    Elite = Pool[1]                  /*  Preserve best player          */
    Crossover Pool
    for m = 1 to Mutation_Rate do  Mutate_Pool
    Pool[Pool_Size] = elite          /* Insert elite back in Pool      */
    Display and store results
}
```

Figure 2 - The first GA used for a RR draughts tournament

```
----------------------------------------------------------------Generation: 50
Fitness Piece King   Mob  KMob  Back  Cent  KCent Adv1  DBLSq  Near Exch  MOVE
Range:  1200  2000   140   140   140   140   140   140   660     5   140   240
------------------------------------------------------------------------------
    30   989  1590    74    53    65    22   127    83   385     1    79   203
    29   904  1590    74     5    65    33   127   137   385     1    79   123
    28   904  1590    74   107    65    33   127    92   385     2    79    98
    28   904  1590   133    76    65    35   127    83   473     1    79    98
    26   904  1692    74   126    73    33   127   137   385     1    79    98
==============================================================================
```

Figure 3 - Sample top-five weights from the final (50th) generation pool

For a pool size of 30, there are approximately 900 games per generation thus giving about 45,000 games played per experiment. It should be noted that each individual game takes about 2 seconds and thus each experiment takes approximately 30 hours on a Pentium 90. The search limit for the look-ahead by DRAFT5 was set to two moves (plus hot-pursuit) so that these experiments would be feasible in a reasonable time-scale. Due to the time consuming nature of the processing a pool size of more than 30 was not considered.

A sample top-five from the final (50th) generation pool is shown in figure 3. In passing, it can be seen that the board feature weights show signs of convergence. Firstly, perhaps the most encouraging result from these initial experiments was the fact that the King Weight was always approximately 1.5 times the (ordinary) Piece Weight. This is the generally accepted ratio as given in many draughts books for human players and indeed was the ratio which was used by Samuel in his studies. This means that three (ordinary) pieces will be exchanged for two kings, if by so doing some positional advantage is obtained.

Secondly, the values for the lesser weights such as mobility, centre control, cramping and advancement were found to be very similar to those determined by years of fine tuning using human opponents playing DRAFT5. Many years of testing and tuning of DRAFT5 against volunteer opponents had produced considerable success and best results with similar values for most of these positional board-feature weights.

Thirdly, DRAFT5 played some games against human opponents using these GA-calculated weights with some success. This ‘variant’ of the draughts program DRAFT5 with automatically determined weight settings is referred to as DRAFT5-GA throughout this paper.

5 Measuring the Improvement of DRAFT5-GA

In the area of draughts and chess the success and ability of a human player (and indeed a program) is usually given by measuring the results obtained against other players (or programs) in tournaments. This characteristic of ranking players and the work of Donnelly et al with the game of GO[Donnelly94] suggested the following method of determining a more *absolute* measure of the improvement of DRAFT5-GA obtained while learning using the GA described above in figure 2.

In a second set of experiments, the winner of each draughts tournament held during each generation of the GA is preserved in a finalists pool. At the end of 50 generations, these generation winners compete against each other in a final all-play-all tournament to determine whether the GA has been successful in improving the playing ability of DRAFT5-GA.

The graph in figure 4 shows the number of wins by generation winners plotted against generation number. From the graph in figure 4, it can be seen that there is a general trend of slight improvement, but that this is superimposed with a lot of local variability. As with the work of Donnelly et al, it is felt that one of the main factors affecting performance is probably the difficulty of obtaining reliable fitness information when using the win/loss results against the other versions of DRAFT5-GA. This is again partly due to the large number of drawn games being very common in draughts.

6 DRAFT5 versus DRAFT5-GA

A third set of experiments were conducted using a slight variation of the basic notion described in the previous section. In this set of experiments however, the winner of each generation was entered into a 40 game match against the original, hand-tuned DRAFT5. The original DRAFT5 uses the same set of features as DRAFT5-GA with

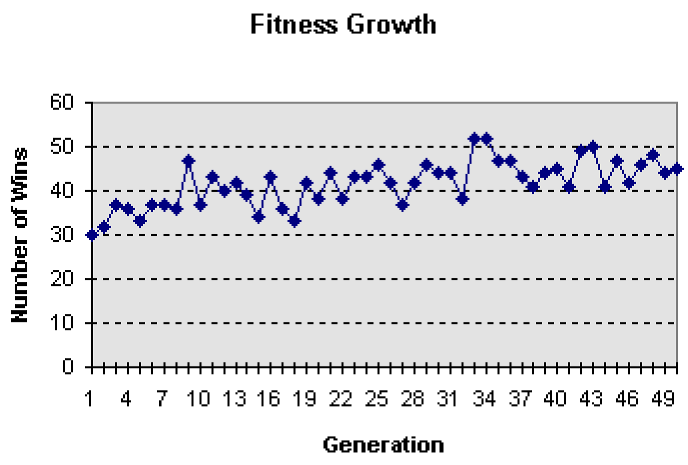

Figure 4 - Graph of relative fitness improvement

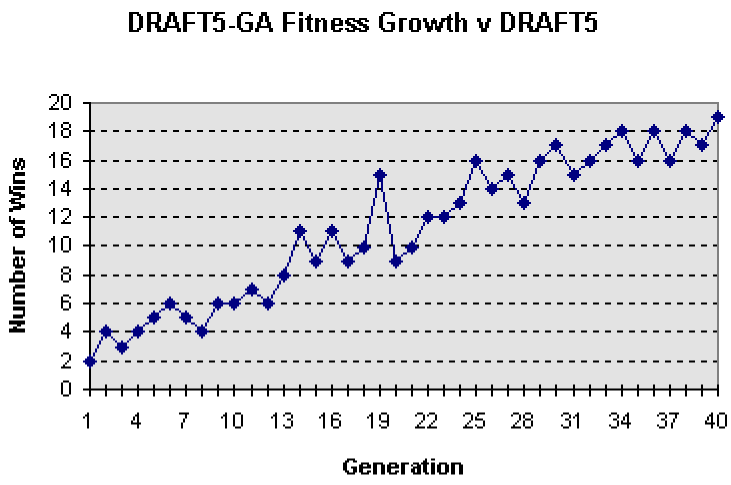

Figure 5 - Fitness Improvement of DRAFT5-GAs v DRAFT5

hand-tuned values for the feature weights. Over the years DRAFT5 is known to play well from its performance against many good human players and other draughts programs. The results of this set of experiments were also quite promising and are shown in figure 5. Again there is also a general trend of improvement with some superimposed local variability.

7. **Conclusions**

The major conclusion that can be drawn from this work is that a relatively unsophisticated GA can determine a good set of board-evaluation weights to play draughts without the addition of any domain specific information, such as specialist crossover or inoculating the pool with known good starting points[Surrey96]. To date, this system has demonstrated a lack of sensitivity to the selection mechanism employed.

Acknowledgements

The authors would like to thank the anonymous reviewers for their constructive comments.

References

[Belasco73] A. Belasco, “Chess and Draughts - How to Play Scientifically”, Foulsham, 1973

[Berliner73] H.J. Berliner, “Some Necessary Conditions for a Master Chess Program”, Third International Joint Conference on Artificial Intelligence, Stanford, CA, 1973.

[Chisholm76] K.J. Chisholm, “DRAFT5 - A Learning Draughts Program”, B.Sc. Project Report, University of Edinburgh, 1976.

[Donnelly94] P. Donnelly, P. Corr & D. Crookes, “Evolving Go Playing Strategy in Neural Networks”, AISB Workshop on Evolutionary Computing, Leeds, England, 1994.

[Fortman82] R. Fortman, “Basic Checkers”, Available from the American Checkers Federation, 1982.

[Goldberg89] D.E. Goldberg, “Genetic Algorithms in Search, Optimization & Machine Learning”, Addison-Wesley, 1989.

[Holland75] J.H. Holland, “Adaption in Natural and Artificial systems”, University of Michigan Press, Ann Arbor, 1975.

[Knuth75] D.E. Knuth & R.W. Moore, “An Analysis of Alpha-Beta Pruning”, Artificial Intelligence, Volume 6, No. 4, 1975.

[Lake94] R. Lake, J. Schaeffer & P. Lu, “Solving Large Retrograde-Analysis Problems Using a Network of Workstations”, Advances in Computer Chess VII, (Ed. H.J. van den Herik et al), University of Limberg, Netherlands, pages 135-162, 1994.

[Levy91] D. Levy & M. Newborn, “How Computers Play Chess”, Computer Science Press, 1991.

[Rosin95] C.D Rosin & R.K. Belew, “Methods for Competitive Co-evolution: Finding Opponents Worth Beating”, Proceedings of the Sixth International Conference on Genetic Algorithms, pp 373-380. Morgan Kaufmann, 1995.

[Samuel59] A.L. Samuel, “Some Studies in Machine Learning Using the Game of Checkers”, IBM Journal of Research and Development, Vol. 3, No. 3, 1959.

[Samuel67] A.L. Samuel, “Some Studies in Machine Learning Using the Game of Checkers II - Recent Progress”, IBM Journal of Research and Development, Vol. 11, No. 6, 1967.

[Schaeffer92] J. Schaeffer, J. Culbertson, B.K. Treloar, P. Lu & D. Szafron, “A World Championship Calibre Checkers Program”, Artificial Intelligence, Vol. 53, pp 273-289, 1992.

[Schaeffer93] J. Schaeffer, N. Treloar, P. Lu & R. Lake, “Man Verses Machine for the World Checkers Championship”, AI Magazine, Vol. 4, No. 2, pp 28-35, 1993.

[Shannon50a] C.E. Shannon, “Programming a Digital Computer for Playing Chess”, Philosophy Magazine, Vol. 41, 1950.

[Shannon50b] C.E. Shannon, “Automatic Chess Player”, Scientific American, Vol. 182, No. 48, 1950.

[Surrey96] P.D. Surrey & N.J. Radcliffe, “Inoculation to Initialise Evolutionary Search”, AISB Workshop on Evolutionary Computing, University of Sussex, 1996.

[Turing53] A.M. Turing, “Digital Computers Applied to Games”, Faster Than Thought (Ed. B.V. Bowden), pp 186-310, 1953.