An Adaptive Evolutionary Algorithm Based on Typical Chess Problems for Tuning a Chess Evaluation Function

Eduardo Vázquez-Fernández
Computer Science Department
Centro de Investigación y de Estudios Avanzados del IPN
México, DF
eduardovf@hotmail.com

Carlos A. Coello Coello
Computer Science Department
Centro de Investigación y de Estudios Avanzados del IPN
México, DF
ccoello@cs.cinvestav.mx

Feliú D. Sagols Troncoso
Mathematics Department
Centro de Investigación y de Estudios Avanzados del IPN
México, DF
fsagols@math.cinvestav.edu.mx

ABSTRACT

This paper presents a method for adjusting weights of the evaluation function of a chess engine. Such an adjustment is carried out through an evolutionary algorithm which adopts a mechanism that selects the virtual players (individuals in the population) that have the highest number of problems properly solved from a database of typical chess problems. This method has the advantage that we only mutate those weights involved in the solution of the current problem. Furthermore, the mutation mechanism is adapted through the number of problems solved by each virtual player. Our results indicate that the material values obtained by our approach are similar to the values known from chess theory. Additionally, we also show that, using the approach proposed here, the strength of our chess engine is increased in 335 points.

Categories and Subject Descriptors

I.2 [**Artificial Intelligence**]: Learning

General Terms

Algorithms

Keywords

Evolutionary algorithm, chess program, evaluation function

1. OUR CHESS ENGINE

To carry out our experiments, we implemented a chess engine with the following characteristics: election of movements through the alpha-beta algorithm [3], search depth of 4 ply, stabilization of positions through the Quiescence algorithm that takes into account the exchange of material and king's checks, and use of hash tables [1]. During the evolutionary process, our chess program evaluates a given position for a particular side, with the following expression:

$$eval = \sum_{i=1}^{r} M_i + \sum_{i=1}^{s} P_i \qquad (1)$$

GECCO'11, July 12–16, 2011, Dublin, Ireland.
ACM 978-1-4503-0690-4/11/07.

where: r is the number of pieces of one side in particular, regardless of the king, s is the number of pieces of one side in particular, M_i represents the material value for piece i (it is a static value) and P_i represents the positional value for piece i (it is a dynamic value that depends on the features of a position).
The positional value P_j of the piece j is given by:

$$P_j = \sum_{i=1}^{4} X_{j,i} * F_{j,i} \qquad (2)$$

where: j can be a king, rook, knight or pawn, $X_{j,i}$ is the weight of factor $F_{j,i}$, $F_{king,1}$ is the sum of material values of pieces that defend their king, $F_{king,2}$ is the sum of material values of pieces that attack the king, $F_{king,3}$ is 1 if and only if the king is castled, $F_{king,4}$ is the number of pawns that protect their king, $F_{rook,1}$ is the mobility of the rook, $F_{rook,2}$ is 1 if and only if the rook is in an open column, $F_{rook,3}$ is 1 if and only if the rook is in the seventh row, $F_{rook,4}$ is 1 if and only if there are two rooks in the seventh row, $F_{knight,1}$ is the mobility of the knight, $F_{knight,2}$ is 1 if and only if the knight is in the periphery of the board, $F_{knight,3}$ is 1 if and only if the knight is defended by a pawn, $F_{knight,4}$ is 1 if and only if the knight cannot be evicted by an enemy pawn, $F_{pawn,1}$ is 1 if and only if the pawn is doubled, $F_{pawn,2}$ is 1 if and only if the pawn is isolated, $F_{pawn,3}$ is 1 if and only if the pawn is central (i.e., if it is in $c4$, $c5$, $d4$, $d5$, $e4$, $e5$, $f4$ or $f5$ square), $F_{pawn,4}$ is 1 if and only if the pawn is passed.

The purpose of this paper is to tune the weights of equation 1 and 2 using evolutionary programming [2] and a database of chess problems [5] (in all the previous works that we found in which the authors use evolutionary algorithms, this decision is based on the outcome of the game).

2. OUR PROPOSED APPROACH

Algorithm 1 shows the evolutionary algorithm used to adjust the weights of our chess engine. The initial population of our evolutionary algorithm consisted of $N = 8$ virtual players whose weights were randomly initialized within their allowable bounds using a uniform distribution. These bounds were defined by a chess expert. The left and right bounds for the material values M_i for the knight and bishop, the rook, and the queen were $[200, 400]$, $[400, 600]$ and $[800, 1000]$, respectively. The left and right bounds for the weight $X_{j,i}$ for the king, rook, knight and pawn were $[-200, 200]$. The number of training chess problems $numP$ was set to 40.

Algorithm 1 EvolutionaryAlgorithm()

```
 1: $P \leftarrow chooseProblems(S)$;
 2: for each problem $p$ in $P$ do
 3:    for each virtual player $i$ do
 4:       foundSolution[i] $\leftarrow$ sol[i] $\leftarrow$ 0;
 5:    end for
 6:    establishWeightsToMutate();
 7:    setPosition($p$);
 8:    $m \leftarrow solution(p)$;
 9:    $g \leftarrow 0$;
10:    while $g{+}{+} < Gmax$ do
11:       for each virtual player $i$ do
12:          $n \leftarrow nextMovement(i)$;
13:          if $m == n$ then
14:             foundSolution[i] = TRUE;
15:             sol[i]++;
16:          end if
17:       end for
18:       if allProblemsFoundSolution()==TRUE then
19:          break;
20:       end if
21:       selection();
22:       mutation();
23:    end while
24: end for
```

Each parent that passes to the following generation is entitled to mutate itself in order to produce an offspring. The values that are mutated are the weights M_i and X_i from the equations 1, 2. In our implementation, we adopted Michalewicz's non-uniform mutation operator [4]. Under this operator, $V_k^{'}$, the mutated weight, is obtained using the following equation:

$$V_k^{'} = \begin{cases} V_k + \Delta(t, UB - V_k) & \text{if R=TRUE} \\ V_k - \Delta(t, V_k - LB) & \text{if R=FALSE} \end{cases} \tag{3}$$

where the weight $V_k \in [LB, UB]$ and $R = flip(0.5)$. The function $flip(p)$ returns TRUE with a probability p. The expression for $\Delta(t, y)$ is:

$$\Delta(t, y) = y * (1 - r^{(1-t/T)^b}) * (1 - sol[i]/numP) \tag{4}$$

where r is a uniformly distributed random number between 0 and 1, T is the maximum number of generations and b is a user-defined parameter; in our case, $b = 2$. The probability of mutation is adapted through the evolutionary process by the term $(1 - sol[i]/numP)$, where $sol[i]$ denotes the number of problems solved by the virtual player i, and $numP$ denotes the number of problems chosen from the database for adjusting the weights.

3. EXPERIMENTAL RESULTS

3.1 Tuning weights

In our experiments, we tuned the weights M_i and $X_{j,i}$ of equations 1 and 2, respectively. If, after mutation, any weight falls, either to the left or to the right of the allowable range $[X_{i,low}, X_{i,high}]$, then its value is set to $X_{i,low}$, or to $X_{i,high}$, respectively. At the beginning of the evolutionary process for run #30 the (average material value, standard deviation) were $(306.7, 55.4)$, $(285.3, 49.4)$, $(506.8, 39.5)$, and $(887.4, 34.9)$ for M_{knight}, M_{bishop}, M_{rook}, and M_{queen}, respectively. At the end of the evolutionary process for run #30 the (average material value, standard deviation) were $(259.9, 0.0)$, $(278.2, 0.0)$, $(472.4, 24.4)$, and $(913.6, 16.9)$ for M_{knight}, M_{bishop}, M_{rook}, and M_{queen}, respectively. Note that the material values of the pieces are nearby their theoretical values [6], which are: 300, 330, 500 and 900 for the knight, bishop, rook and queen, respectively. The material value for the pawn is always 100. The weights of a virtual player without evolution (corresponding to run #30) were used to play 10 games against a chess player with 1820 rating points. In this case, our chess engine recorded a rating of 1425 points. The same chess player also conducted ten games against a virtual player with adjusted weights. In this case, our chess engine recorded 1760 rating points.

4. CONCLUSIONS AND FUTURE WORK

The method that has been presented in this paper is based on an evolutionary algorithm whose selection mechanism gives priority to the virtual players who had properly solved more problems selected from a database. Our proposed method allows to mutate only those weights involved in the current problem, preventing incorrect mutations that can lead to incorrect values for future evaluations of board positions. Moreover, our method adapts the mutation rate based on the number of problems that have been solved for each virtual player. The material values of the pieces obtained by our approach are similar to the values known from chess theory, and the strength of the chess engine was increased from 1425 to 1760 points during the evolutionary process (a chess player intermediates). As part of our future work, we plan to run our evolutionary algorithm with a larger number of chess problems (both tactical and positional in nature) and with a larger number of weights (mainly, the weights associated with the positional values of the bishop and the queen) in our evaluation function in order to increase the rating of our chess engine as much as we can.

Acknowledgements

The first author acknowledges support from CINVESTAV-IPN, CONACyT and the National Polytechnical Institute (IPN) to pursue graduate studies at the Computer Science Department of CINVESTAV-IPN. The second author acknowledges support from CONACyT project no. 103570.

5. REFERENCES

[1] D. Beal and M. C. Smith. Multiple probes of transposition tables. *ICCA Journal*, 19(4):227–233, 1996.

[2] L. J. Fogel. *Artificial Intelligence through Simulated Evolution*. John Wiley, New York, 1966.

[3] D. E. Knuth and R. W. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6(4):293–326, 1975.

[4] Z. Michalewicz. *Genetic Algorithms + Data Structures = Evolution Programs*. Springer-Verlag, second edition, 1996.

[5] F. Reinfeld. *One Thousand and One Winning Chess Sacrifices and Combinations*. Wilshire Book Company, 1969.

[6] C. Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 7(41):256–275, 1950.