Nine Men’s Morris: Evaluation Functions

Simona-Alexandra PETCU, Stefan HOLBAN
Computer Science and Engineering, Politehnica University of Timisoara

***Abstract*—Since the beginning of artificial intelligence, programmers have been aiming to create the unbeatable computer player. Most of the progress was done in classic board games. For the majority, this task has been achieved, like DEEP-BLUE for chess. For others, improvements still need to be made. For example, Othello programmers are still searching for the most efficient estimator model. A board game programmer has two main tasks: to perform search tree optimizations and to find an efficient evaluation function (estimator). This paper is connected to the second task and its goal is to present the steps that were made, in order to design an appropriate estimator for the game “Nine Men’s Morris”.**

***Index Terms*— estimator model, evaluation functions, game tournament, genetic evolution**

I. INTRODUCTION

Game artificial intelligence is one of several ways to create the illusion of intelligent behaviors in machines. Although the search tree algorithms go from the very simple to the most complex, they all have a common key point, the evaluation of possible configurations. In order to choose the most appropriate move, an evaluation function is required, in order to turn a table configuration into a relevant numeric value. This function computes the value of each configuration; therefore most of the game strategy lies within it.

Human gamers use various strategies and follow certain patterns depending on the game phase and the board configuration. These patterns are of great help for those who can turn them into an advantage. If the computer is not aware of certain patterns, it can be fooled very easily even by a non-experienced human player. It can play clever in most of the situations but gets caught easy in others. The algorithm doesn’t “see” further than the programmer tells it to. A good game developer will know the game strategies and will also know how to make the PC Player aware of those strategies.

“Nine Men’s Morris” is a classic 2 player board game. Although it is less complex and also less popular than chess, it is a good example to experience the artificial intelligence algorithms for game playing. After having developed a Java “Nine Man’s Morris” game I am working on improving the evaluation functions of the estimator, for the PC player.

Starting from an estimator model composed of three weighted sum evaluation functions (one for each phase of the game), I have used a competitive technique that submits estimators to tournament, in order to find the best ones. For the evolution of the estimators, a genetic crossover algorithm is used.

The next section is an overview of the game application, and the algorithms used. The third section presents the estimator problems, decisions and the steps, to improve the evaluation functions. Section four presents the conclusions.

II. THE APPLICATION

The game application is developed in Java, using OOP techniques. The user can choose between 3 types of games (human vs. human, human vs. PC and PC vs. PC). The user can choose a profile for the PC Player, including the level, play mode (Offence or Defense) and the algorithm used to compute the moves. In the PC Player context, I made use of two search algorithms: Alfa-Beta and one extension of this algorithm, Minimal Window Search. The difference between them consists in the way they perform the tree search.

The algorithms generate the states’ tree from the current position, on a certain number of levels (the depth of the algorithm). When the last level is reached, each configuration (each leaf) must be evaluated and is assigned a value. The two players are seen as MAX and MIN. A configuration of higher value is considered good for MAX, and one of lower value is better for MIN. (This can be also translated as positive value for MAX, and negative value for MIN). In other words, MAX player will move towards configurations of high value, while the MIN player will try to go towards the lowest value ones.

A board configuration must be evaluated from both players’ perspective. The idea is to compute, using the same method, two estimators: one for MAX and one for MIN. The final estimator will be computed as the substraction of the two: TotalEst = Est_MAX – Est_MIN. If TotalEst is a positive value it means that MAX has an advantage in that configuration. If TotalEst is negative, the advantage is on MIN’s side.

The estimator’s method of computation is crucial for the success of the game.

III. THE ESTIMATOR

“Nine Men’s Morris” game has three important phases (clearly delimited by the rules, but also by the techniques and strategies used by players). In the first phase, players alternatively place their pieces on the board. If a player closes a morris he will grab an opponent’s piece. The way the pieces are placed is crucial for the rest of the game. This is why the experienced players, in this phase, concentrate upon strategically placing the pieces rather than on closing a morris. The second phase starts when all the pieces were placed. Players move the pieces one step at a time, close morrises and grab pieces. The best configuration is considered to be the double morris. For each player, the third phase starts when he is left with only 3 pieces. Now he must act very carefully, but the advantage of this phase is that each of his pieces can be moved anywhere on the board (not just one step as in phase 2).

1. Evaluation functions, relations and coefficients

One of the estimator structural decisions was to take in consideration the phase of the game, for each player, when computing the value for a table configuration. So, the estimator is composed of *three evaluation functions* based on weighted sum. Computing the value consists of analyzing the current configuration, finding certain relations between the pieces and give points. Each relation will have a different weight (coefficient). For example, if a morris is created (closed) when moving a piece, that configuration will receive some extra 30 points, in comparison to a similar configuration that does not have a closed morris. A closed morris is one of the basic relations between pieces and must be taken into consideration when evaluating the configuration.

These relations are chosen on the basis of human players' experience. They represent the relations and associations that a human player would look for when playing this game. So, they are based on the common knowledge of the game and on strategies of expert human players.

Table 1. presents the relations chosen for the game. Each game phase has its own set of relations, although some relations were found relevant for more than one phase.

Table 1 – Game configuration relations.

Phase 1 relations	Phase 2 relations	Phase 3 relations
R1: *Closed morris* R2: *Morrises number* R3: *Number of blocked opp. pieces* R4: *Pieces number* R5: *Number of 2 pieces configurations* R6: *Number of 3 pieces configurations*	R1: *Closed morris* R2: *Morrises number* R3: *Number of blocked opp. pieces* R4: *Pieces number* R5: *Opened morris* R6: *Double morris* R7: *Winning configuration*	R1: *2 pieces configurations* R2: *3 pieces configurations* R3: *Closed morris* R4: *Winning configuration*

Most of the relations are clear and do not need further explanations. The ones that might not be so suggestive are described in the following lines. A "2 pieces configuration" refers to having 2 pieces on one line (and the other place empty, so morris possibility). A "3 pieces configuration" offers the opportunity to close a morris in two places so the opponent cannot stop you. These refer to the first stage (placing pieces) and also to the final phase. "Closed morris" refers to a new closed morris, that offers the advantage to grab one opponent piece, while "morrises number" refers to the total number of morrises one player has got.

Figure 1. shows the structure of the estimator, with an evaluation function for each phase. Rx from phase k represents Relation no. x from phase no. k (see table 1). Cfx refers to the coefficient corresponding to Rx. In all three phases we have a total of 17 relations therefore 17 different coefficients.

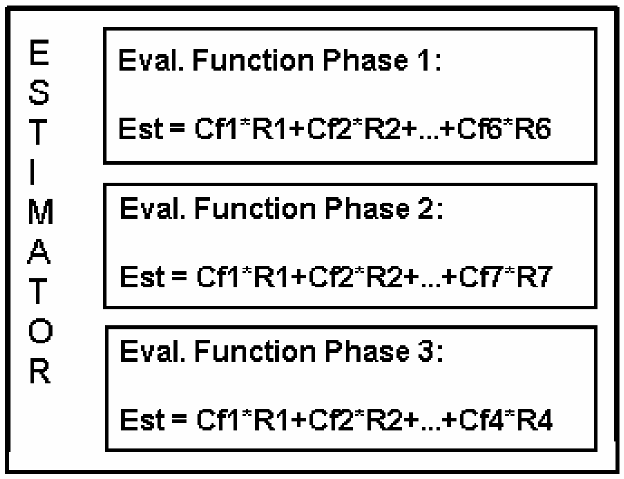

ESTIMATOR
Eval. Function Phase 1:
Est = Cf1*R1+Cf2*R2+...+Cf6*R6
Eval. Function Phase 2:
Est = Cf1*R1+Cf2*R2+...+Cf7*R7
Eval. Function Phase 3:
Est = Cf1*R1+Cf2*R2+...+Cf4*R4

Figure 1. Estimator model.

In order to compute the Est value for one player, the program simply checks the game phase for that player and applies the correspondent function.

When a configuration is generated, it is analyzed and we will know exactly (for each player) how many morrises has he got, how many pieces he has got left, and so on.

Let's analyze, as a short example, the tree in Figure 2 with only a few states (table configurations):

Let's suppose that the game is in the second stage and "Square" is to move.

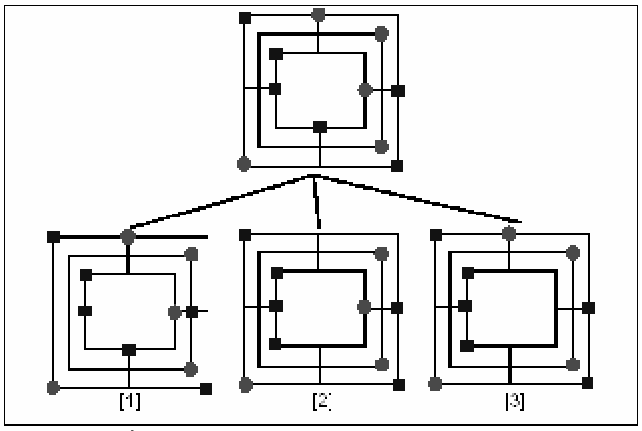


Figure 2. Configurations tree.

From the three presented moves, number [1] is a good choice because it will prevent the opponent ("Circle") from closing a morris. Configuration [2] has a closed morris and an opponent piece is grabbed. But, this advantage is superficial, as in the next move "Circle" will probably close his morris too, and obtain a piece in return. Number [3] seems to be the best alternative for the moment. The morris is closed, and by grabbing the right piece (one from the opened morris), the opponent is also prevented from closing a morris in the next move. All these logical appreciations must be translated into functions and coefficients for the PC Player.

For the basic version of the game, I developed two Estimators: one for "Defense mode" and one for "Offence mode". In order to make the PC play better, the relations of the functions remain unchanged, but the coefficients are varied. I tuned the coefficients manually, based on trials. Defense is based more on conserving the pieces and blocking opponent's pieces while Offence based more on closing morrises and double morris structure. To be based on certain relations translates into growing the coefficients' value of those relations. To make a relation count less, simply lower its coefficient or even give it the 0 value.

3.2 Choosing the coefficients

At first, choosing the coefficients for Defense and Offence estimators was mostly done based on assumptions and human vs. PC tests. But, the number of games that one can play becomes irrelevant as there are so many possible combinations. One logical way to get closer to a good estimation function would be to let different estimators compete with each other. At this point, we shall leave the relations out, as they are computed for each configuration. If one player closes a morris, the "closed morris" relation is set from 0 to 1, if a piece is blocked, the "blocked pieces counter" realtion is incremented and so on.

The discussion shall be made upon the coefficients. One estimator must be a set of 17 coefficients (the total number for all 3 stages). After a great number of games, we shall have some "winner estimators" that are most likely to play good also against a human player.

So, for this game, in order to find the most appropriate estimator I have implemented a 2 stages "Tournament", shown in the workflow image below (Figure 3).

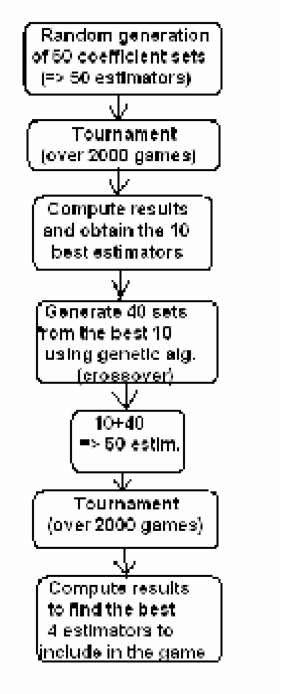

Random generation of 50 coefficient sets (=> 50 estimators)
Tournament (over 2000 games)
Compute results and obtain the 10 best estimators
Generate 40 sets from the best 10 using genetic alg. (crossover)
10+40 => 50 estim.
Tournament (over 2000 games)
Compute results to find the best 4 estimators to include in the game

Figure 3. Estimator selection workflow.

I've started with 50 estimators (50 sets of 17 coefficients each). There sets were generated randomly, but within certain bounds. I used a function called *generate(int a, int b)* that returns a random value between a and b.

For example for a winning configuration the coefficient should be a random value between 900 and 1200, while for a closed morris between 20 and 40. It should be even smaller for a blocked opponent piece or for the number of pieces.

In the tournament, each Ei!=Ej estimator plays 2 games: Ei vs. Ej and Ej vs. Ei (because sometimes it is important which player starts the game). The tournament had as input a file containing all 50 sets, and the output consisted of one file for each played match (containing info about the player estimators, the winner and the moves list). After a tournament of over 2000 games, the resulting files were processed. Only about 75% of the games were useful information, as the others ended up in a "Tie" (each game was automatically stopped after 85 moves if no estimator had won by then). Each estimator received 5 points/ each game won. The output was a file containing the estimators and scores in descending order. The best 10 were selected.

In order to make the estimators evolve, I applied a simple one-crossover genetic algorithm, like in Figure 4, obtaining another 40 combined estimators.

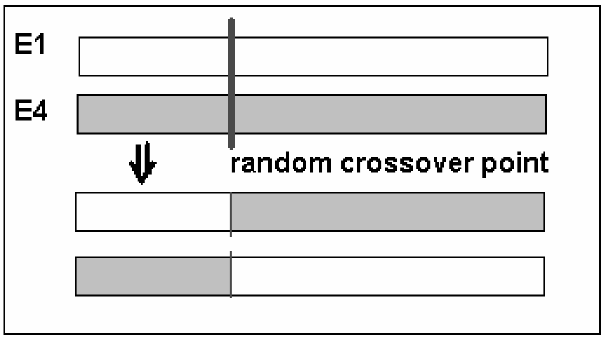


Figure 4. One point crossover.

In a genetic algorithm the population involved must have a fitness function, showing the crossover capability for each individual. Having to obtain 40 estimators from the best 10, I choose the following structure: [(E0; 10), (E1; 7),(E2; 5), (E3; 5), (E4; 4), (E5; 3), (E6; 2), (E7; 2), (E8; 1), (E9; 1)]. The list contains each estimator (E0 is the best) and the number of crossovers it will take part in. This means that, randomly, the estimator with the highest score (E0) will take part in 10 crossovers; E1 will take be part in just 7, and so on. The last, E9, will take part in only one exchange. The participants to an information exchange (crossover) will be randomly established, but in the certain bounds. For example if E4 has already attended 4 crossovers, even if it is randomly chosen again, it is not allowed to participate and another estimator will be selected. These conditions are all included in the genetic algorithm that generates the new sets. The crossover point is another variable randomly chosen, for each crossover.

All 50 (the best 10 plus the derived 40) were then again submitted to a tournament, with the same procedures as the first. This time, the ones with the best 3 scores were chosen to be integrated in the game and to be tested in human vs. PC games, to see if they are indeed efficient.

Table 1 presents the results. The best 4 estimators (the third and forth have equal score, the same number of won matches): E32 (crossover between E1 and E7), E34 (crossover between E1 and E2), E30(crossover between E1 and E4), and E46 (crossover between E3 and E7).

It is interesting to observe that the best configurations are crossovers of E1, which is not the best estimator resulted from the first tournament (E0), but the second best.

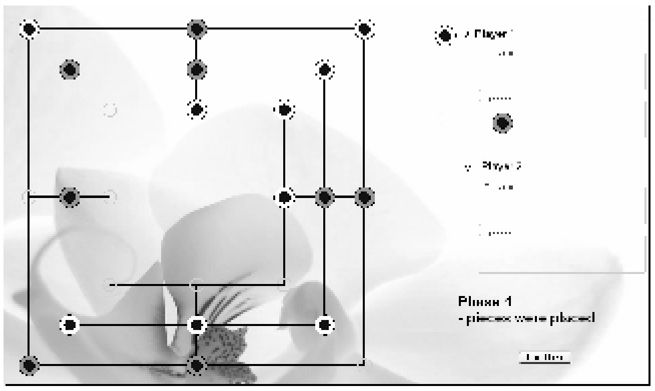
Table 2 – Coefficient sets of the best estimators.

	Phase 1	Phase 2	Phase 3
E32	18 26 1 6 12 7	14 43 10 8 7 42 1086	10 1 16 1190
E34	18 26 1 6 21 7	42 28 16 8 24 19 949	23 18 5 1096
E30	18 26 1 6 12 7	14 43 10 1 30 40 958	6 32 7 1041
E46	14 37 4 14 20 2	16 43 11 8 7 42 1086	10 1 16 1190

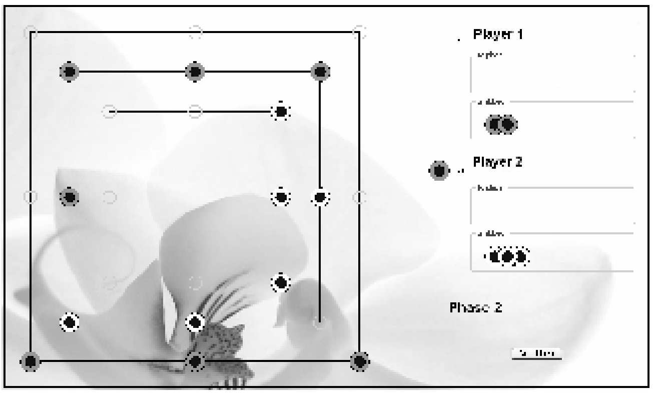
The first three estimators from the table share common parts of the coefficients' list. Those are inherited from E1. At different points (crossover points) these estimators go on different directions. E32 and E46 also have a common part (towards the end of the sequence), from E7.

For an overall impression of the game design and tournament, Figure 5. presents captures from a match, where E0 (the best estimator from the first tournament) was beat by E32 (the best estimator from the second tournament).

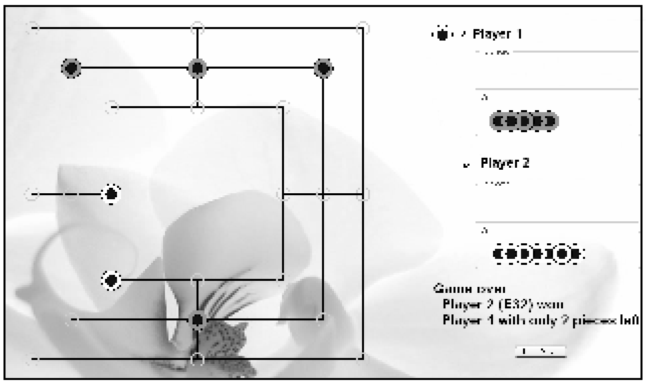
After the first phase (fig. 5a), the advantage is of E0's side (Player 1), having already grabbed an opponent piece. But, as the game continues, they both manage to have two morris structures (big. 5b). E32 (Player 2) manages to grab more pieces from E0 and so, it manages to win the match leaving the E0 with only two pieces on the board.



5a



5b



Game over
Player 2 (E32) won
Player 1 with only 2 pieces left

5c

Figure 5. Nine Men's Morris Match.

IV. CONCLUSIONS

The resulting best estimators from PC vs. PC tournaments are able to play good matches also against human players and are better than the initial Defense and Offence estimators, although these are not based on defense or offense approaches.

Although E0 was the best estimator in the first tournament, in the second one, the system has evolved towards the E1 model. Many values and genetic algorithm parameters were generated randomly, to obtain diversity. There could be performed several experiments like this one, the process of selection can be started over again for another group of estimators, and another "family" will be obtained. Combining the best estimators could lead to an even better PC Player.

REFERENCES

[1] Bruno Bouzy, Tristan Cazenave, Computer Go: An AI oriented survey, 2000.

[2] Michael Bruno, Improving heuristic mini-max search by supervised learning, 2002

[3] Stuart Russel, Peter Norwig, Game Playing(pg.122-149), Artificial Intelligence – A Modern Approach, Prentice Hall, 2003.

[4] Iordache Ciprian-Doru, Leonte Bogdan Nicolae, Țîmpu Radu Cristian, Teoria Jocurilor, 2002-2003.

[5] George F Luger, Artificial Intelligence – Structures an Strategies for Complex Problem Solving, Fifth Edition, Addison-Wesley, 2005.

[6] Victor Aguirregabiria,Pedro Mira, A Genetic Algorithm for the Structural Estimation of Games with Multiple Equilibria, 2005.

[7] Xavier Llora, Kumara Sastry, Frances Alias, David E. Goldberg, Michael Welge, Analyzing Active Interactive Genetic Algorithms using Visual Analytics, 2006.