Ensemble Evolution of Checkers Players with Knowledge of Opening, Middle and Endgame

Kyung-Joong Kim and Sung-Bae Cho

Department of Computer Science, Yonsei University
134 Shinchon-dong, Sudaemoon-ku, Seoul 120-749 South Korea
{kjkim, sbcho}@cs.yonsei.ac.kr

Abstract. In this paper, we argue that the insertion of domain knowledge into ensemble of diverse evolutionary checkers can produce improved strategies and reduce evolution time by restricting search space. The evolutionary approach for game is different from the traditional one that exploits knowledge of the opening, middle, and endgame stages, so that it is not sometimes efficient to evolve simple heuristic that is found easily by humans because it is based purely on a bottom-up style of construction. In this paper, we have proposed the systematic insertion of opening knowledge and an endgame database into the framework of evolutionary checkers. Also, common knowledge, the combination of diverse strategies is better than the single best one, is inserted into the middle stage and is implemented using crowding algorithm and a strategy combination scheme. Experimental results show that the proposed method is promising for generating better strategies.

1 Introduction

Incorporating a priori knowledge, such as expert knowledge, meta-heuristics, human preferences, and most importantly domain knowledge discovered during evolutionary search, into evolutionary algorithms has gained increasing interest in recent years [1]. In this paper, we propose a method for systematically inserting expert knowledge into an evolutionary checkers framework at the opening, middle, and endgame stages. In the opening stage, openings defined by the American Checkers Federation (ACF) are used. In previous work, we have used speciation techniques to search for diverse strategies that embody different styles of game play and have combined them using voting for higher performance [2]. This idea comes from the common knowledge that the combination of diverse well-playing strategies can defeat the best one because they can complement each other for higher performance. Finally, we have used an endgame database from Chinook, the first man-machine checkers champion. Figure 1 explains the conceptual framework of the proposed method.

The most important idea is the systematical integration of three domain knowledge (opening DB, middle stage knowledge and endgame DB). The middle stage knowledge is coming from the Korean event in the game of Go. In 2003, Internet site TYGEM (http://www.tygem.co.kr) held a many-to-one style game between Hoon Hyun Cho, one of the greatest go players, and 3000 amateur players. The winner of

Q. Yang and G. Webb (Eds.): PRICAI 2006, LNAI 4099, pp. 950 – 954, 2006.

the game was Cho. After the game, he said that it was a very difficult game because there was no obvious mistake of amateur players. Speciation algorithm for evolutionary checkers is adopted for an implementation of the knowledge.

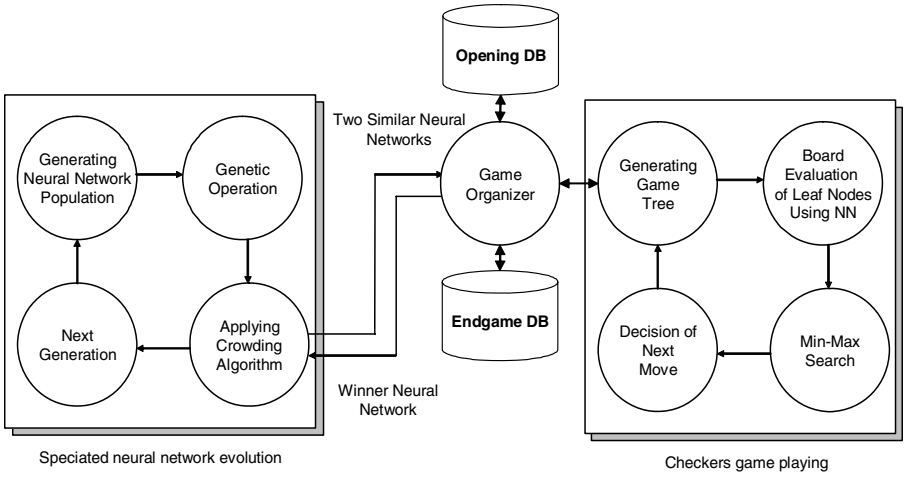

Fig. 1. Conceptual diagram of the proposed method

2 Incorporating Knowledge into Evolutionary Checkers

2.1 Opening Stage

The opening move is the most important opportunity to defeat an expert player because trivial mistakes in the opening can lead to an early loss. The first move in checkers is played by red and there are seven choices (9-13, 9-14, 10-14, 10-15, 11-15, 11-16, and 12-16). Usually, 11-15 is the best move for red but there are many other alternatives. They are described with specific names, such as Edinburgh, Double Corner, Denny, Kelso, Old Faithful, Bristol, and Dundee, respectively. For each choice, there are many well established more sequences which range in length from 2 to 10. The longest sequence is described as the White Doctor: 11-16, 22-18, 10-14, 25-22, 8-11, 24-20, 16-19, 23-16, 14-23, 26-19. Careful analysis over decades of tournament play has proven the usefulness or fairness of the opening sequences. Initial sequences are decided by the opening book until the move is out of the book. Each player chooses their opening randomly and the seven first choices have the same probability to be selected as an opening.

2.2 Evolutionary Speciated Checkers

Following Fogel [3], a checkers board is represented by a vector of length 32 and components in the vector could have a value of $\{-K, -1, 0, +1, +K\}$, where K is the

value assigned for a king, 1 is the value for a regular checker, and 0 represents an empty square. For reflecting spatial features of the board configuration, sub-boards of the board are used as an input. One board can have 36 3×3 sub-boards, 25 4×4 sub-boards, 16 5×5 sub-boards, 9 6×6 sub-boards, 4 7×7 sub-boards and 1 8×8 sub-board. 91 sub-boards are used as an input to the feed-forward neural network. The sign of the value indicates whether or not the piece belongs to the player or the opponent. The closer the output of the network is to 1.0, the better the position is. Similarly, the closer the output is to -1.0, the worse the board.

The architecture of the network is fixed and only the weights can be adjusted by evolution. Each individual in the population represents a neural network (weights and biases) that is used to evaluate the quality of the board configuration. Additionally, each neural network has the value of K and self-adaptive parameters for weights and biases. An offspring $P_i', i = 1,..., p$ for each parent $P_i, i = 1.,, , p$ is created by

$$\sigma_i'(j) = \sigma_i(j)\exp(\tau N_j(0,1)), \quad j = 1,..., N_w$$
$$w_i'(j) = w_i(j) + \sigma_i'(j)N_j(0,1), \quad j = 1,..., N_w$$

where N_W is the number of weights and biases in the neural network (here this is 5046), $\tau = 1/\sqrt{2\sqrt{N_w}} = 0.0839$, and $N_j(0,1)$ is the standard Gaussian random variable resampled for every j. In fitness evaluation, each individual chooses five opponents from a population pool and plays games with the players. Fitness increases by 1 for a win while the fitness of an opponent decreases by 2 for a loss. In a draw, the fitness values of both players remain the same. After all the games are played, the fitness values of all players are determined.

In this paper, we utilize a crowding algorithm [4], a popular form of speciation algorithm, for searching for diverse neural networks. In this algorithm, one neural network is selected from two similar individuals based on the result of game played between them (usually, a crowding algorithm uses their fitness but in this case, we cannot use fitness because of the dynamic property of fitness landscape). A crowding algorithm is one of the representative speciation methods that attempt to discover diverse species in a search space. The distance between two neural networks is calculated by using Euclidean distance between their weights. To discover clusters of individuals in the population at the last generation with arbitrary shape, density-based clustering methods have been used. DBSCAN (Density-based Spatial Clustering of Applications with Noise) is one of the algorithms [5]. Representative players from each cluster are chosen by tournament of all players in the same cluster. Moves of combined players are determined using a simple voting of the representative players. It picks the move that has the greatest number of votes. If there is no clear winner, one of the moves that have the greatest votes is selected randomly.

2.3 Endgame Stage

The estimated quality of the board is calculated using the evolved neural networks to evaluate the leaf nodes of the tree with min-max algorithm. If the value of *f* (estimated goodness of the next moves) is not reliable, we refer to the domain specific knowledge and revise *f*. The decision rule for querying the domain knowledge must

be defined previously as follows. **IF** (f<0.75 and f>0.25) or (f<-0.25 and f>-0.75) **THEN** querying the domain knowledge.

3 Experimental Results

The non-speciated evolutionary algorithm uses a population size of 15 and limits the run to 60 generations. The speciated evolutionary algorithm sets the population size to 15 and generations to 60. The mutation rate is 0.01 and crossover rate is 1.0. The number of leagues (it is used to select the best player from each species) is 5 (5 means that each player selects 5 players from species randomly and the competition results are used for the selection).

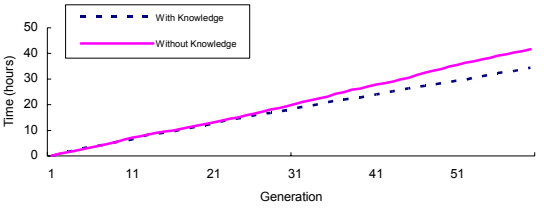

Fig. 2. Comparison of running time (Simple evolution)

The Chinook endgame DB (2~6 pieces) is used for revision when the estimated value from the neural network is between 0.25 and 0.75 or between -0.25 and -0.75. Time analysis indicates that the evolution with knowledge takes much less time than that without knowledge in simple evolution (Figure 2). This means that the insertion of knowledge within a limited scope can accelerate the speed of evolutionary algorithm because it can reduce computational requirement for finding optimal endgame sequence by using endgame DB. Table 1 summarizes the competition results between the best individual in the evolution with knowledge and the best individual in the evolution without knowledge for each generation. The knowledge incorporation model can perform better than the one without knowledge. Table 2 shows the competition results in the speciated evolution. Table 3 shows the effect of the stored knowledge (opening and endgame DB) in speciation.

Table 1. Experimental results on opening and endgame knowledge incorporation (Win/Lose/Draw) for simple evolution. Evolution with the stored knowledge performs better than that without the knowledge. (Op=Opening knowledge, SGA=Simple GA, E=Endgame knowledge).

Op+SGA+E	**SGA**	Generations				
		1~14	15~29	30~44	45~59	Total
Red	White	5/0/10	3/3/9	3/0/12	5/3/7	16/6/38
White	Red	4/3/8	4/2/9	5/4/6	4/2/9	17/11/32

Table 2. Experimental results on opening and endgame knowledge incorporation (Win/Lose/Draw) for speciated evolution. Evolution with the stored knowledge performs better than that without the knowledge. (S=Speciation).

Op+S+E	**Speciated**	Generations				
		1~14	15~29	30~44	45~59	Total
Red	White	5/1/9	4/3/8	6/0/9	8/2/5	23/6/31
White	Red	7/3/5	5/2/8	8/4/3	6/2/7	26/11/23

Table 3. The competition results between the speciated players using both opening and endgame DB and the speciated player with one of the knowledge

Op+S+E	**Op+S**	**Total**
Red	White	6/2/7
White	Red	8/4/3
Op+S+E	**S+E**	**Total**
Red	White	5/5/5
White	Red	4/5/6
Op+S	**S+E**	**Total**
Red	White	3/6/6
White	Red	2/7/6

4 Conclusion and Future Work

The final conclusion of the experiment is SGA < Speciated < Op+S < S+E ≈ Op+S+E (SGA < Speciated is from the results of [2]). The effect of opening knowledge is not so big because they have only the limited sequences. The limited opening knowledge can prevent the player from making a big mistake but it is not much useful when the opponent chooses a move that is not included in the opening sequence. Multiple diverse neural networks can perform better than the single best one but there is always problem of combination and averaging may not work. As a future work, sophisticated combination method should be explored for better performance.

References

1. Jin, Y.: Knowledge Incorporation in Evolutionary Computation, Springer (2004)
2. Kim, K.-J. and Cho, S.-B.: Evolving speciated checkers players with crowding algorithm. Proc. of the 2002 Congress on Evolutionary Computation (2002) 407-412
3. Fogel, D.B.: Blondie24: Playing at the Edge of AI. Morgan Kaufmann (2001)
4. Mahfoud, S. W.: Niching methods. Handbook of Evolutionary Computation, C6.1, IOP Publishing and Oxford University Press, (1997)
5. Ester, M., Kriegel, H.-P., Sander J. and Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise. Knowledge Discovery and Data Mining (1996) 226-231