Introducing Individual and Social Learning Into Evolutionary *Checkers*

Belal Al-Khateeb and Graham Kendall, *Senior Member, IEEE*

***Abstract*—In recent years, much research attention has been paid to evolving self-learning game players. Fogel's Blondie24 is just one demonstration of a real success in this field and it has inspired many other scientists. In this paper, evolutionary neural networks, evolved via an evolution strategy, are employed to evolve game-playing strategies for the game of *Checkers*. In addition, we introduce an individual and social learning mechanism into the learning phase of this evolutionary *Checkers* system. The best player obtained is tested against an implementation of an evolutionary *Checkers* program, and also against a player, which has been evolved within a round robin tournament. The results are promising and demonstrate that using individual and social learning enhances the learning process of the evolutionary *Checkers* system and produces a superior player compared to what was previously possible.**

***Index Terms*—Artificial neural networks, *Checkers*, evolutionary algorithms, individual and social learning.**

I. Introduction

INTEREST in designing automated computer game-playing programs dates back to at least the 1950s [1], [2], [42], [43] and this interest has not diminished in recent years. Blondie24 is an evolutionary algorithm that was presented by Fogel and Chellapilla [3], [4] and is capable of playing the game of *Checkers*. One of the main aims of their investigation was to minimize (if not eliminate) the amount of expert (human) knowledge that was input to the algorithm. By only using the number, type, and positions of pieces on the *Checkers* board, along with minimax, the evolutionary program utilizes feedforward artificial neural networks to evaluate alternative board positions. The core feature in the design of Blondie24 was to make the program learn, through self-play, how to play *Checkers*. This is an alternative approach to writing an algorithm with the help of human experts and of utilizing an evaluation function to judge the quality of a given board position. The architecture of Blondie24 is shown in Fig. 1 [3].

The motivation of this work is inspired by the success of Blondie24. The experiments described in this paper are based on an evolutionary algorithm that was used as the basis for Blondie24, but we hypothesize that the introduction of an individual and social learning mechanism will evolve a superior player. It was not our intention to produce a world *Checkers* champion. Rather, we want to study the effectiveness of introducing individual learning and social learning as a machine learning approach to game playing.

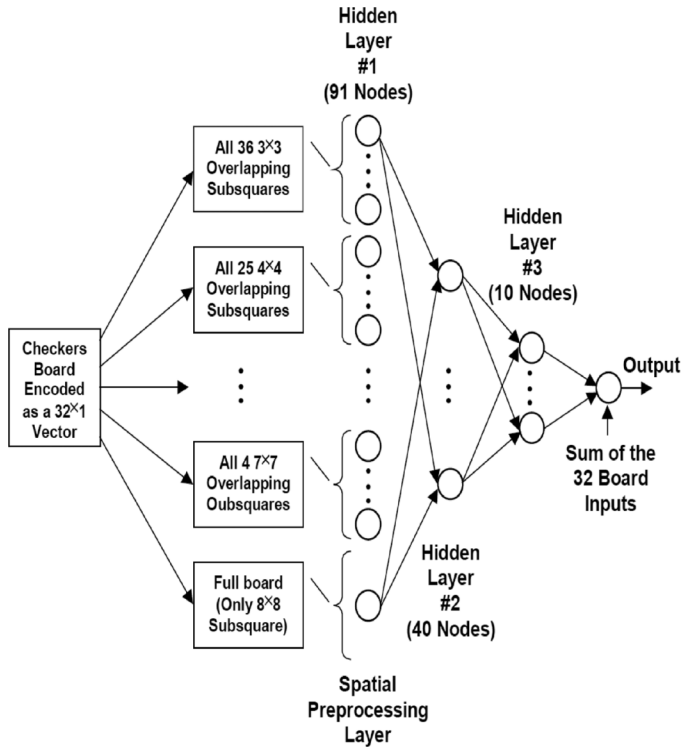

Fig. 1. Blondie24 architecture [3].

The rest of the paper is organized as follows. In Section II, related work is presented. The experimental setup is described in Section III. Section IV presents our results and Section V concludes the paper together with some suggestions for future work.

II. Background

A. Samuel's Seminal Work

In 1954, Samuel developed a *Checkers* player in an attempt to demonstrate that a computer program could improve by playing against itself. Samuel's program adjusted weights for 39 features [2], [5], [42]–[44]. Samuel used a form of what is now called "reinforcement learning" (more details about reinforcement learning can be found in [6]–[8]) to adjust these features, instead of tuning them by hand. Samuel discovered that the most important feature was the piece difference and the remaining 38 features (including capacity for advancement, control of the

Manuscript received November 26, 2010; revised April 08, 2011, August 09, 2011, and February 03, 2012; accepted June 13, 2012. Date of publication July 18, 2012; date of current version December 11, 2012.

B. Al-Khateeb is with the College of Computer, Al-Anbar University, Ramadi 31, Iraq (e-mail: belal@computer-college.org).

G. Kendall is with the School of Computer Science, The University of Nottingham, Nottingham NG7 2RD, U.K. (e-mail: gxk@cs.nott.ac.uk).

Digital Object Identifier 10.1109/TCIAIG.2012.2209424

1943-068X/$31.00 © 2012 IEEE

center of the board, threat of fork, etc.) varied in their importance. Due to memory limitations, Samuel only used 16 of the 38 features in his evaluation function, swapping between them to include the remaining 22, which he called term replacement [2], [3], [5], [42]–[44].

Two evaluation functions (alpha and beta) were used to determine the weights for the features. At the start, both alpha and beta have the same weight for every feature. Alpha weights were modified during the execution of the algorithm. Beta values remained static. The process gave an appropriate weight to each feature when summed together. Each leaf node in the game tree was evaluated using this evaluation function. This process represents one of the first attempts to use heuristic search methods in searching for the best next move in a game tree. Samuel [2], [42], [43] used minimax with three-ply search and a procedure called *rote learning*. This procedure was responsible for storing the evaluation of different board positions in a lookup table for fast retrieval (look-ahead and memorization). Samuel [5], [44] improved the minimax search with alpha–beta pruning that incorporated a supervised learning technique to allow the program to learn how to select the best parameters to be calculated in the evaluation function.

In July 1962, Samuel's program played against Robert Nealey, described (incorrectly) as a state champion (he was not to earn that title for another four years). Samuel's program defeated Nealey. At that time, it was considered a great success and a significant achievement in machine learning. In fact, this was the only win that Samuel's program managed against Nealey, or any other player, and there is some controversy about how strong a player Nealey really was. Samuel claimed that his program focused on the problem of having a machine learning program, rather than being told how to play, but in fact, he used 39 features (although he wanted to get away from that requirement), which some would argue is utilizing human knowledge. However, the historical importance of this work cannot be underestimated as it set the challenge, which Fogel was later to accept, and to answer.

B. Chinook

In 1989, Schaeffer and his colleagues at the University of Alberta (Edmonton, AB, Canada), designed a *Checkers* program called Chinook [9]. [10], which later became the world champion at *Checkers*. Schaeffer's initial motivation was to *solve* the game. However, this was a challenging goal as there are approximately 5×10^{20} different positions to evaluate [10].

A further motivation was to produce the world's best *Checkers* player. This was done by using an evaluation function, which comprises several features, all of them based on human expertise, including grand masters. The main feature in Chinook's evaluation function is the piece count, where each piece on the board takes 100 points. The next most important feature is the king, which takes a value that is greater than a regular checker by 30%, except when the king is trapped (a trapped king cannot move because it will be taken by the opponent), when it takes the same value as a regular checker. Another feature that is important to Chinook's evaluation function is the *runaway* checker (a clear path for a checker to become a king, without any obstacles), which takes a value of 50 points in addition to its previous value, and subtracts three points for each move that is required to advance the checker to be a king. There are other additional features that are included in the evaluation function, including the "*turn*," "*mobile kings*," and the "*dog hole*" (a checker that is trapped by its opponent and cannot be moved). Each one of those features was assigned a different weight indicating its importance.

The summation of each term provided an overall assessment of the board for that particular game state, which enabled different game states to be compared. Initially, Schaeffer gave initial values to the weights and then hand tuned them when he found an error (e.g., an obviously incorrect move being made) or when a Chinook move led to a losing position.

Chinook also utilized opening and end game databases to further enhance its ability. Initially, Chinook's opening game database contained 4000 sequences. Later, it contained more than 40 000. The end game database contained all the possibilities that led to a win, a draw, or a loss, for a given number of pieces left on the board. The final version of Chinook's end game database contained all six-piece end sequences, allowing it to play perfectly from these positions.

In 1989, Chinook, with a four-piece end game database [11], won the computer Olympiad. Later, with its final six-piece end game database, together with its evaluation function modified by a *fudge* factor [10], [12], it finished in second place to Marion Tinsley (recognized as the best *Checkers* player who ever played the game) at the U.S. National Checkers Championship held in 1990. After a further sequence of matches in 1994 between Chinook and Tinsley, Chinook became the world man machine *Checkers* champion (after Tinsley's resignation due to health problems; he died the following year) [10]. In 1996, Chinook retired with rating at 2814.

The building of the open/end game databases ultimately led Schaeffer to achieve his initial motivation (solving the game of *Checkers*) [13]. Perfect play by both sides leads to a draw.

C. Blondie24

Blondie24 represents an attempt to design a computer *Checkers* program, injecting as little expert knowledge as possible [3], [4], [14]–[17]. Evolutionary neural networks were used as a self-learning computer program. The neural network used for a particular player provided the evaluation function for a given board position. Evolutionary pressure caused these networks, which acted randomly initially (as their weights were initialized randomly), to gradually improve over time. The final network was able to beat the majority ($>99\%$) of human players registered on www.zone.com at that time. Blondie24 represents a significant achievement, particularly in machine learning and artificial intelligence. Although Blondie24 does not play at the level of Chinook [10], this was not the objective of the research; rather, it was to answer the challenges set by Samuel [1], [2], [42], [43], as well as to answer Newell and Simon (two early AI pioneers) who said that progress in this area would not be made without addressing the credit assignment problem. The major difference between Blondie24 and other traditional game-playing programs is in the employment of the evaluation function [14], [15]. In traditional game-playing programs, the evaluation function usually consists of important

features drawn from human experts. The weighting of these features is altered using hand tuning, whereas, in Blondie24, the evaluation function is an artificial neural network that only knows the number of pieces on the board, the type of each piece, and their positions. The neural network is not preinjected with any other knowledge that experienced players would have.

The following algorithm represents Blondie24 [4], [17].

1) Initialize a random population of 30 neural networks (strategies) $P_i = 1, \ldots, 30$ sampled uniformly $[-0.2, 0.2]$ for the weights and biases.
2) Each strategy has an associated self-adaptive parameter vector $s_i = 1, \ldots, 30$ initialized to 0.05.
3) Each neural network plays against five other neural networks selected randomly from the population.
4) For each game, each competing player receives a score of $+1$ for a win, 0 for a draw, and -2 for a loss.
5) Games are played until either one side wins, or until 100 moves are made by both sides, in which case a draw is declared.
6) After completing all games, the 15 strategies that have the highest scores are selected as parents and retained for the next generation. Those parents are then mutated to create another 15 offspring using the following equations. For each parent $P_i, i = 1, \ldots, 15$, an offspring P_i' was created by

$$s_i(j) = s_i(j) \exp\left(tN_j(0,1)\right), \qquad j = 1, \ldots, N_w \quad (1)$$

$$w_i(j) = w_i(j) + s_i(j)N_j(0,1), \qquad j = 1, \ldots, N_w \quad (2)$$

where N_w is the number of weights and biases in the neural network (here, this is 5046), $t = 1/\sqrt{2\sqrt{Nw}} = 0.0839$, and $N_j(0,1)$ is a standard Gaussian random variable resample for every j.
7) Repeat steps 3–6 for 840 generations (this number was an arbitrary choice in the implementation of Blondie24).

Blondie24 represents a milestone in evolutionary learning, but the design did not allow for the end product to learn any further after the self-play stage. That is, learning was only exercised in the evolution phase and no learning took place in the playing phase, when it faced human opponents. This makes Blondie24 incapable of adapting itself when interacting with human players. Harley comments on this fact in his book review [18]:

> "An interesting point is that the end product which looks intelligent is Blondie, yet she is not in fact the intelligence. Like the individual wasp, Blondie is fixed in her responses. If she played a million games, she would not be an iota smarter. In this sense, she is like Deep Blue. . . . Perhaps a better example of intelligence would be . . . a human, who can adapt her behavior to any number of new challenges."

It would be interesting, and conceptually easy (though challenging logistically), to enable Blondie24 to continue to learn, after the self-play stage, when it starts playing against humans. This is certainly a research direction worth pursuing but, we suspect, that the logistical challenges make it difficult to carry out in practice.

The creation of Blondie24 is to be considered as a learning process (achieving Samuel's challenge [5], [44]) but the current version of Blondie24 is unable to learn from its environment [19] (i.e., playing against humans). Another point of note is that in step 3 of the algorithm, the strategies do not all play the same number of games because, by chance, some will be selected as opponents more often than others. Al-Khateeb and Kendall [20] enhanced Blondie24 by introducing a round robin tournament, instead of randomly choosing the opponents. We draw on this work in this paper (see Section IV).

In order to investigate the effects of introducing individual learning and social learning to an evolutionary *Checkers* system, we first implemented an evolutionary *Checkers* player (which we refer to as C_0). Our implementation is based on the same structure and architecture that Fogel utilized in Blondie24.

D. Two-Move Ballot

When the world's best players play the game of *Checkers*, it often ends in a draw. To overcome this, and make the games more competitive, the two-move ballot is used.

This was introduced in the 1870s [10]. The first two moves (each side's first move) are randomly chosen. There are 49 possibilities to play in this way, but research has shown that six of these moves (openings) are unbalanced, as it will give an advantage to one side over the other. Therefore, only 43 of the 49 available moves are considered. At the start of the game, a card is randomly chosen indicating which of the 43 openings is to be played. The original game, with no forced opening moves, is called go-as-you-please (GAYP).

E. Bayeselo

The Elo rating system [21] was originally used for *Chess*, but it is now used for many other games (e.g., football; see http://www.eloratings.net/system.html). Each player (or team) is given an initial rating, and after playing each other, their rating changes as a function of their current rating and whether they win, lose or draw.

Elo says that the expected result of a game is a function of the difference in rating between two players. For example

$$E = 1/\left(1 + 10\text{^}(D/400)\right)$$

where E is the expected result and D is the rating difference between two players.

Elo assumes that there is a single value that can represent the player's strength, and therefore, the expected result can be determined according to the above formula. One of the problems with Elo is that you cannot take a set of game results and produce a ranking of all those players.

In this paper, we use a modified version of Elo, called Bayeselo [22], to compare various evolved players. Bayeselo finds a likelihood of superiority (LOS), using a minorization–maximization algorithm [23]. Tables I and II show an example of Bayeselo estimates and LOS. Software can be

TABLE I
BAYESELO RATINGS EXAMPLE

Rank	Name	Elo	+	-
1	P_1	598	156	123
2	P_2	345	45	98

TABLE II
BAYESELO LOS EXAMPLE

	P_1	P_2
P_1	-	87
P_2	12	-

downloaded [22], which enables all of the necessary calculations to be made.

Table I shows the Elo rating for two players after they have played a number of games (not necessarily against just each other). The $+/-$ columns shows the true rating at a 95% confidence level (you can change the confidence level in the program, but we leave it at 95% for all the results reported in this paper). Taking P_1 as an example, its Elo rating is 598 and its true value is between (598 − 123) 475 and (598 + 156) 754, at the 95% confidence level.

The values in Table II show how the supremacy of players is reported using Bayeselo. You can treat the figures as percentages, representing the LOS. Table II shows that there is an 87% likelihood that P_1 is stronger than P_2 and a 12% likelihood that P_2 is stronger than P_1.

In this paper, we use the LOS as the main statistical measure of the superiority of one player over that of another. If this value is around 50%, we assume that the players are equal. If the value is above 90%, we assume that the players are statistically different, at the 95% confidence level.

It is also worth noting (for the purposes of reproducibility) that we change the *advantage* parameter to zero when running Bayeselo. The default is +32, which is used for *Chess* (representing the advantage white has) but this is not applicable to *Checkers* so we set this parameter value to zero.

III. INDIVIDUAL LEARNING AND SOCIAL LEARNING

Humans, when developing strategies to defeat other humans, use a variety of techniques. For example, humans can improve their strategy by themselves or through learning from the experience of competing with other humans. Developing their own strategies based on a copy of a better player model is another technique utilized by humans. In other words, humans can learn through both individual and social learning.

In [24], social learning is stated as "learning from others," while [30] defines individual learning and social learning as: "With individual learning an agent learns exclusively on the basis of his own experience, whereas population or social learners base themselves on the experience of other players as well." In general, social learning can be defined as learning indirectly from the experiences of others (as opposed to one's own experiences). In competitive learning [25], in order to survive to the next generation, all the players will play against each other. The sources of inspiration for our work can be found in [19] and [26]–[29], where a simulated stock market used coevolving neural networks (evolved through a process of individual learning and social learning). Agent-based computational economics is by far the most common use of social learning research [26], [30].

In this work, we define individual learning as the mechanism by which a player chooses five other players to play against [step 3) in the algorithm], in order to try and improve itself. This might still be considered social learning (as the agent is playing against another agent, although they are learning from their own experience), but we use this terminology as each player is given the opportunity to try and improve itself, and it also differentiates it from the social learning mechanism, which is when an agent is able to draw from a collective pool of experience.

Individual learning and social learning are utilized in two stages. The player will undertake individual learning by playing against five other players. After a certain time has elapsed, we enter a social learning phase when players are able to learn from each other.

In social learning, the player is given the chance to copy a strategy from a pool of previously good strategies, or to generate a new strategy to replace its current strategy. For completeness, according to our definition, Blondie24 can be considered as individual learning.

Best strategies from the population are retained in a social pool. This pool is made available to those players which are not performing well. In this respect, it closely resembles a hall of fame [25], where the progress of learning is tested against a panel of all the best evolved players at every generation. There are two reasons to save the best players in every generation. One is to contribute genetic material to future generations. The second reason is for the purpose of testing. Hall of fame has been applied to many games such as *Nim* and *3-D Tic-Tac-Toe* and has been shown to be successful as it significantly improved the coevolutionary performance for those games.

In social learning, the player has the opportunity to choose to replace their existing strategy with another one selected from the social pool, in the hope that the selected strategy is better than their current strategy. All strategies in the social pool have their own score, updated over time. The activities in social learning are as follows:

1) rank the players in descending order;
2) copy the best player or players (if more than one) to the social pool;
3) for the rest of the players, there are two possibilities:
   a) if the player is satisfied with his current strategy (based on the current score), retain that strategy;
   b) if the player is not satisfied with his current strategy, three alternatives are available:
      i) copy a strategy from the pool;
      ii) create a new random strategy;
      iii) retain their current strategy.

When considering social learning, it is interesting to compare it with the island model in evolutionary computation. In an island model, each individual in a subpopulation evolves independently [29], [31]. Moreover, the best player from a subpopulation can migrate to another subpopulation, if and only if, it is the better strategy. However, there is no creation of a new strategy

in the subpopulation. In social learning, as mentioned above, the individual players have the opportunity to copy a better strategy, retain their current strategy, or generate a new random strategy.

The individual and social learning mechanism is also different from the case-injected genetic algorithm (CIGAR) [32], [33] that combines genetic algorithms with case-based reasoning to play a computer strategy game. CIGAR works by injecting the best strategies (players) obtained from past games into the population of a genetic algorithm in order to try and produce better players. This can be done along with a suitable representation. Results demonstrate that case injection can produce superior players.

Cultural algorithms are also different from individual and social learning mechanisms, since cultural algorithms [34], [35] are models of evolutionary learning that are set to emulate cultural evolutionary processes. Two levels of evolution constitute a cultural algorithm, namely, microevolution in a population space and macroevolution in a belief space. Utilizing an acceptance function, the experiences of individuals in the population space are employed to create problem solving knowledge, which is then stored in the belief space. The knowledge is manipulated by the belief space and this subsequently guides the evolution of the population space through an influence function. A fraud detection system was designed by Sternberg and Reynolds [36] who used a cultural-algorithm-based evolutionary learning approach to learn about the behavior of a commercial-rule-based system for detecting fraud. The acquired knowledge in the belief space of the cultural algorithm is then used to re-engineer the fraud detection system. Another application of cultural algorithms is in modeling the evolution of complex social systems [37], [38]. Furthermore, the application of cultural algorithms for function optimization problems in dynamic environments has been described by Reynolds and Saleem [39], [41] and Reynolds and Peng [40]. In their experiments, the dynamic environment is modeled as a 2-D plane on which four cones of varying heights and slopes are haphazardly positioned. At certain generations, the four cones change their locations on the plane hence the location of the optimum solution is constantly changing. When applied to the problem of finding the new optima in dynamic environments, Reynolds and Saleem [39] demonstrated that a cultural algorithm is superior (in the set of experiments that were carried out) compared to an evolutionary algorithm with only one single-level evolution. Reynolds and Peng [40] discuss how the learning of knowledge in the belief space warrants the adaptability of cultural algorithms. Reynolds and Saleem [41] further examine the contributions of various types of knowledge from the belief space in piloting the quest for the best solutions in both deceptive and nondeceptive environments.

A. An Application of the Framework to Checkers

Our hypothesis is that the introduction of social learning into an evolutionary *Checkers* system will provide a richer environment for learning. The players outside the social pool are called individual players, all of which attempt to develop their own strategy. At certain times, the best players are drawn from the social pool to replace poorly performing individual players.

In our experiments, we have made some modifications to the algorithm described in [19] in order to investigate how to increase the number of players in the social pool, thus, producing a larger number of strategies that can be copied by individual players.

We propose two phases. The first will use individual learning, with the best players being copied to the social pool after every M generations. In the second phase, social learning occurs every N generations (see below for justifications for choosing the values for M and N in this paper). Therefore, in comparison to [19], we copy strategies to the social pool more often. The algorithm for our experiments is as follows.

---

1) Initialize a random population of 30 neural networks (players) sampled uniformly $[-0.2, 0.2]$ for the weights.
2) Each player has its associated self-adaptive parameter, initialized to 0.05.
3) Initialize M (frequency of individual learning) and N (frequency of social learning).
4) For each player in the current population, randomly choose five players to play against.
5) For each game, the player receives a score of $+1$ for a win, 0 for a draw, and -2 for a loss.
6) Games are played until either side wins, or until 100 moves are made by both sides, in which case a draw is declared.
7) If the generation number is exactly divisible by M and not by N, then:
   — select the best player(s) with the highest score (if two or more players have equal scores, we will select all those players) and copy them to the social pool;
   — select the best 15 players and mutate them to get 15 offspring using (1) and (2).
8) If the generation number is exactly divisible by N, then for all players i do:
   — normalize the individual scores (values between 0 and 1) for all the players using

$$V_i = \frac{(X_i - \mathrm{Min})}{(\mathrm{Max} - \mathrm{Min})} \tag{3}$$

   where V_i is the normalized value for player i, Min and Max are the lowest and highest scores in the current population among all players, and X_i is the score of player i before being normalized;
   — if the normalized value is 1 and the player is not using a strategy drawn from the pool, then publish the strategy into the pool;
   — if the normalized value is 1 and the player is using a strategy drawn from the pool, then do not publish the strategy into the pool but update the strategy's score in the pool;
   — for the rest of the players, there are two cases:
      1) if the normalized value is between 1 and 0.9, then the player is satisfied with his current strategy and retains it:

TABLE III
NUMBER OF WINS FOR THE EVOLVED C_0 PLAYERS (ROW PLAYER) OUT OF 774 GAMES

Player	1	2	3	4	5	6	7	8	9	10
1	-	22	25	20	19	17	23	24	22	20
2	20	-	20	22	24	21	20	21	23	24
3	21	19	-	20	21	19	18	23	21	22
4	25	23	18	-	20	24	20	24	18	19
5	24	20	23	19	-	18	20	22	22	21
6	21	24	22	23	20	-	22	20	23	22
7	24	18	20	21	21	19	-	19	18	20
8	25	17	19	24	20	20	17	-	22	23
9	21	22	21	18	22	24	20	24	-	22
10	19	22	24	20	21	22	20	21	24	-

- if the normalized value is less than 0.9, then the player is not satisfied with his current strategy; the player has three options:
  a) with 1/3 probability, replace the current strategy by copying a new strategy from the pool; roulette wheel selection is used to select the new strategy from the pool;
  b) with 1/3 probability, replace the current strategy by creating a new random strategy;
  c) with 1/3 probability, retain the current strategy.

9) If the generation number is not exactly divisible by M or N, then:
   — select the 15 best players and mutate them to get 15 offspring using (1) and (2).
10) Repeat steps 4)–9) for K generations or for specified time.

B. Conducted Experiments

Three experiments were carried out. The first experiment is to ensure that C_0 and Blondie24-RR (which is a result of our previous work to enhance Blondie24 obtained by introducing a round robin tournament into Blondie24 [20]) evolve in a consistent way. We evolved ten C_0 players and ten Blondie24-RR players. We compare players of the same type (i.e., C_0 or Blondie24-RR) using the two move ballot (so each player plays all 43 openings, both as red and white) and test if they are statistically different by using Bayeselo. Our aim is to show that evolved players using the same experimental parameters are the same. If we are able to establish that, then we can compare players evolved with different experimental parameters to ascertain whether they are different.

The second experiment is designed to determine the best values for the number of generations to determine where the individual (M) and social (N) phases should occur. This experiment is also used to see the effects of increasing the number of players in the social pool. We chose different values for M and N, selecting values which we believe provide a good tradeoff between individual and social learning and the computational cost of testing different values. We recognize that we may not have established the optimum values but, given the stochastic nature of the system, we are not sure that we could ever make this claim. We believe that the experiments we carry out to establish suitable values for M and N are reasonable. The values chosen for (M, N) are (100, 200), (50, 100), (20, 50), (10, 20), and (5, 10). The players representing these values are referred to as follows:

1) C_{200} a player when $M = 100$ and $N = 200$;
2) C_{100} a player when $M = 50$ and $N = 100$;
3) C_{50} a player when $M = 20$ and $N = 50$;
4) C_{20} a player when $M = 10$ and $N = 20$;
5) C_{10} a player when $M = 5$ and $N = 10$.

In order to provide an additional comparison, we also used a baseline player C_1 ($M = 5$, $N = 10$), which simply takes the best player (line 7 in the algorithm), choosing randomly if there are more than one, and retains only this player in the social pool.

The final experiment investigates the effects of introducing individual learning and social learning for evolutionary *Checkers*. This is done by playing each evolved player in the second experiment against C_0 and against Blondie24-RR and using Bayeselo to test the outcome.

In order to provide fair comparisons, we ran the above algorithms for the same amount of time (about 19 days) that was required to produce C_0. All experiments were run on the same computer (1.86-GHz Intel Core2 processor and 2-GB RAM).

IV. RESULTS

A. Experiment 1: Comparison of Evolved Players Using the Same Experimental Settings

In order to ensure that the learning strategies that are used to construct C_0 and Blondie24-RR are consistent, we evolve ten C_0 players and ten Blondie24-RR players. Tables III–V show the results of playing the ten evolved C_0 players against each other and Tables VI–VIII show the results of playing the ten Blondie24-RR players against each other. We use the two-move ballot, so each player plays 86 games, against nine other players, making a total of 774 games for each player.

The results in Table IV show that the ten C_0 players are statistically the same, as all the scores are around 50%. In addition, all the Elo ratings (and the 95% confidence values) are close. The results in Table V indicate that the LOS between the players is acceptable, as most of the values are between 40% and 60% except in a few cases. For example, player 6 is 73% better than player 8, but we consider this acceptable, in light of the other figures. Based on Tables IV and V, there is no statistical difference between the players. We decided to choose the player with the most number of wins (i.e., player #6) to be our baseline player C_0.

TABLE IV
BAYESIAN ELO RATINGS FOR THE EVOLVED C_0 PLAYERS

Rank	Player	Elo	+	-	Games	Score	Draws
1	6	4	17	18	774	51%	51%
2	2	3	18	18	774	51%	51%
3	4	1	18	17	774	50%	51%
4	5	0	18	17	774	50%	51%
5	9	0	18	18	774	50%	50%
6	7	0	17	17	774	50%	53%
7	10	0	18	18	774	50%	50%
8	3	-3	18	18	774	49%	51%
9	1	-3	18	18	774	49%	49%
10	8	-4	17	18	774	49%	50%

TABLE V
LOS FOR THE EVOLVED C_0 PLAYERS

Player	6	2	4	5	9	7	10	3	1	8
6	-	55	59	62	62	63	63	70	70	73
2	44	-	54	57	57	58	58	65	65	68
4	40	45	-	53	53	54	54	62	62	65
5	37	42	46	-	50	51	51	59	59	62
9	37	42	46	49	-	51	51	59	59	62
7	36	41	45	48	48	-	50	58	58	61
10	36	41	45	48	48	49	-	58	58	61
3	29	34	37	40	40	41	41	-	50	53
1	29	34	37	40	40	41	41	49	-	53
8	26	31	34	37	37	38	38	46	46	-

TABLE VI
NUMBER OF WINS FOR THE EVOLVED BLONDIE24-RR PLAYERS (ROW PLAYER) OUT OF 774 GAMES

Player	1	2	3	4	5	6	7	8	9	10
1	-	22	25	23	22	19	23	24	22	20
2	20	-	20	22	24	21	20	21	23	24
3	24	19	-	20	23	20	19	23	21	22
4	25	23	21	-	20	24	20	24	18	20
5	24	22	23	19	-	20	20	22	22	21
6	21	24	22	23	20	-	22	20	24	24
7	24	18	20	21	21	21	-	21	18	20
8	25	21	22	24	20	20	22	-	22	21
9	21	22	21	18	22	24	20	24	-	25

TABLE VII
BAYESIAN ELO RATINGS FOR THE EVOLVED BLONDIE24-RR PLAYERS

Rank	Player	Elo	+	-	Games	Score	Draws
1	6	2	19	19	774	50%	49%
2	4	2	19	19	774	50%	50%
3	9	1	19	19	774	50%	49%
4	5	0	19	19	774	50%	50%
5	10	0	19	19	774	50%	49%
6	2	0	19	19	774	50%	50%
7	7	-1	19	19	774	50%	52%
8	1	-1	19	19	774	50%	48%
9	3	-2	19	19	774	50%	50%
10	8	-2	19	19	774	50%	48%

The results in Table VII show that the ten evolved Blondie24-RR players are the same, as all the scores are about 50%. The results in Table VIII indicate that the LOS between the players is acceptable, as most of the values are between 40% and 60% except in a few cases. Based on Tables VII and VIII, there is no statistical difference between the players. Like C_0, we use the player with the most number of wins (i.e., player #6) to be our Blondie24-RR baseline player.

B. Experiment 2: Parameter Settings for Individual Learning and Social Learning

To measure the effect of introducing individual learning and social learning into an evolutionary *Checkers* system, a league structure between C_1, C_{200}, C_{100}, C_{50}, C_{20}, and C_{10} was held, in order to determine the best values for M and N. Each player was set to play against all other players by using the two-move

TABLE VIII
LOS FOR THE EVOLVED BLONDIE24-RR PLAYERS

Player	6	4	9	5	10	2	7	1	3	8
6	-	52	54	57	57	57	59	60	62	62
4	47	-	52	55	55	55	57	58	60	60
9	45	47	-	53	53	53	55	56	58	58
5	42	44	46	-	50	50	52	53	55	55
10	42	44	46	49	-	50	52	53	55	55
2	42	44	46	49	49	-	51	53	55	55
7	40	42	44	47	47	48	-	51	53	53
1	39	41	43	46	46	46	48	-	52	52
3	37	39	41	44	44	44	46	47	-	50
8	37	39	41	44	44	44	46	47	49	-

TABLE IX
NUMBER OF WINS (FOR THE ROW PLAYER) OUT OF 430 GAMES

Player	C_1	C_{200}	C_{100}	C_{50}	C_{20}	C_{10}	Σwins
C_1	-	22	14	12	10	8	66
C_{200}	35	-	29	22	16	10	112
C_{100}	39	25	-	21	17	12	114
C_{50}	40	37	26	-	21	18	142
C_{20}	47	41	32	27	-	15	162
C_{10}	59	55	49	41	34	-	238

TABLE X
NUMBER OF WINS FOR THE EVOLVED C_{10} PLAYERS (ROW PLAYER) OUT OF 774 GAMES

Player	1	2	3	4	5	6	7	8	9	10
1	-	24	24	25	22	23	25	24	22	23
2	24	-	23	22	24	23	23	23	25	24
3	24	23	-	23	25	23	23	25	23	22
4	24	25	23	-	23	24	23	24	23	23
5	24	22	25	23	-	23	23	22	22	23
6	23	24	22	25	23	-	22	23	24	24
7	24	23	23	23	23	23	-	23	23	23
8	24	23	22	24	23	23	22	-	22	23
9	23	22	23	23	22	24	23	24	-	24
10	23	24	22	23	23	24	23	25	24	-

TABLE XI
BAYESIAN ELO RATINGS FOR THE EVOLVED C_{10} PLAYERS

Rank	Player	Elo	+	-	Games	Score	Draws
1	3	1	18	18	774	50%	46%
2	10	1	18	18	774	50%	46%
3	4	0	18	18	774	50%	45%
4	2	0	18	18	774	50%	46%
5	7	0	18	18	774	50%	46%
6	6	0	18	18	774	50%	46%
7	9	0	18	18	774	50%	46%
8	1	0	18	18	774	50%	45%
9	5	0	18	18	774	50%	46%
10	8	-2	18	18	774	50%	46%

ballot. We play all of the 43 possible games, both as red and white, giving a total of 86 games. The games were played until either one side won or a draw was declared after 100 moves for each player. The total number of games played was 430. Table IX shows the results.

It is worth mentioning that although each player is the result of a single run, the trends in performance are consistent. For example, the wins versus C_1, C_{200}, C_{100}, C_{50}, C_{20}, and C_{10} are all increasing. That is, although there is uncertainty in how representative each player is of the approach used to create it, the trends suggest that the learning strategy is effective. Also, to make sure that this is the case, we decided to evolve nine more players based on the player with the most number of wins (C_{10}) and do a similar comparison to the one we did for C_0 and Blondie24-RR. Tables X–XII show the results.

Table XI shows that the ten evolved C_{10} players are statistically the same as all the scores are around 50%. The Elo ratings for the players are also very close. The results in Table XII indicate that the LOS between the players is acceptable, as most of the values are between 40% and 60%.

TABLE XII
LOS FOR THE EVOLVED C_{10} PLAYERS

Player	3	10	4	2	7	6	9	1	5	8
3	-	52	53	53	53	54	54	55	55	61
10	47	-	51	51	51	52	52	53	53	59
4	46	48	-	50	50	51	51	52	52	58
2	46	48	49	-	50	51	51	52	52	58
7	46	48	49	49	-	51	51	52	52	58
6	45	47	48	48	48	-	50	51	51	57
9	45	47	48	48	48	49	-	51	51	57
1	44	46	47	47	47	48	48	-	50	56
5	44	46	47	47	47	48	48	49	-	56
8	38	40	41	41	41	42	42	43	43	-

TABLE XIII
BAYESIAN ELO RATINGS FOR THE SOCIAL PLAYERS

Rank	Player	Elo	+	-	Games	Score	Draw
1	C_{10}	119	29	29	430	70%	30%
2	C_{20}	41	27	27	430	57%	40%
3	C_{50}	11	27	27	430	52%	38%
4	C_{100}	-23	27	27	430	46%	39%
5	C_{200}	-46	28	28	430	42%	32%
6	C_1	-102	28	28	430	32%	33%

TABLE XIV
LOS FOR THE SOCIAL PLAYERS

Player	C_{10}	C_{20}	C_{50}	C_{100}	C_{200}	C_1
C_{10}	-	99	99	99	100	100
C_{20}	0	-	95	99	99	99
C_{50}	0	4	-	97	99	99
C_{100}	0	0	2	-	89	99
C_{200}	0	0	0	10	-	99
C_1	0	0	0	0	0	-

TABLE XV
RESULTS WHEN PLAYING $C_1, C_{200}, C_{100}, C_{50}, C_{20}$, AND C_{10} AGAINST C_0 USING THE TWO-MOVE BALLOT

Player	Opponent:C_0		
	Win	Draw	Lose
C_1	20	22	44
C_{200}	27	31	28
C_{100}	30	30	26
C_{50}	40	21	25
C_{20}	44	22	20
C_{10}	51	20	15

TABLE XVI
BAYESIAN ELO RATINGS FOR THE SOCIAL PLAYERS VERSUS C_0

Rank	Player	Elo	+	-	Games	Score	Draw
1	C_{10}	118	66	66	86	71%	23%
2	C_{20}	65	64	64	86	64%	26%
3	C_{50}	29	63	63	86	59%	24%
4	C_{100}	-17	60	60	86	52%	35%
5	C_0	-32	26	26	516	45%	28%
6	C_{200}	-35	60	60	86	49%	36%
7	C_1	-128	64	64	86	36%	26%

Tables XIII and XIV show the results of applying the Bayeselo to the results in Table IX. The results in Tables IX and XIII show that C_{10} received the most wins, and has the highest Elo rank among all the other social players. The results in Table XIV show that C_{10} is superior to other social players by at least 99%. Therefore, we conclude that $M = 5$ and $N = 10$ are the best values to use for individual learning and social learning.

C. Experiment 3: Comparing All Players

We now play each player against C_0 and also against Blondie24-RR by using the two-move ballot, testing the results

TABLE XVII
LOS FOR THE SOCIAL PLAYERS VERSUS C_0

Player	C_{10}	C_{20}	C_{50}	C_{100}	C_0	C_{200}	C_1
C_{10}	-	86	97	99	99	99	99
C_{20}	13	-	78	96	99	98	99
C_{50}	2	21	-	84	97	92	99
C_{100}	0	3	15	-	68	66	99
C_0	0	0	2	31	-	54	99
C_{200}	0	1	7	33	45	-	98
C_1	0	0	0	0	0	1	-

TABLE XVIII
RESULTS WHEN PLAYING $C_1, C_{200}, C_{100}, C_{50}, C_{20}$, AND C_{10} AGAINST BLONDIE24-RR USING THE TWO-MOVE BALLOT

Player	Opponent:Blondie24-RR		
	Win	Draw	Lose
C_1	17	16	53
C_{200}	20	29	37
C_{100}	22	28	36
C_{50}	30	17	39
C_{20}	31	25	30
C_{10}	43	18	25

TABLE XIX
BAYESIAN ELO RATINGS FOR THE SOCIAL PLAYERS VERSUS BLONDIE24-RR

Rank	Player	Elo	+	-	Games	Score	Draw
1	C_{10}	108	64	64	86	60%	21%
2	C_{20}	37	62	62	86	51%	29%
3	Blondie24-RR	33	26	26	516	56%	26%
4	C_{50}	-5	64	64	86	45%	20%
5	C_{100}	-20	61	61	86	42%	33%
6	C_{200}	-31	61	61	86	40%	34%
7	C_1	-122	67	67	86	29%	19%

TABLE XX
LOS FOR THE SOCIAL PLAYERS VERSUS BLONDIE24-RR

Player	C_{10}	C_{20}	Blondie24-RR	C_{50}	C_{100}	C_{200}	C_1
C_{10}	-	94	98	99	99	99	99
C_{20}	5	-	54	82	90	93	99
Blondie24-RR	1	45	-	87	95	98	99
C_{50}	0	17	12	-	63	72	99
C_{100}	0	9	4	36	-	59	98
C_{200}	0	6	1	27	40	-	97
C_1	0	0	0	0	1	2	-

using Bayeselo. We play all of the 43 possible games, both as red and white, giving a total of 86 games. The games were played until either one side won or a draw was declared after 100 moves for each player.

The results for each player $\{C_1, C_{200}, C_{100}, C_{50}, C_{20}, C_{10}\}$ against both C_0 and Blondie24-RR are shown in Tables XV–XX.

Table XXI summarizes the results when playing against C_0 and against Blondie24-RR, using a starting position where all pieces are in their original positions (i.e., no two-move ballot).

According to the results in Tables XV–XX, it is not recommended to use a social pool with only one player, as both C_0 and Blondie24-RR have a 99% level of superiority over C_1 (Tables XVII and XX). Also there is no point using $M = 100$ and $N = 200$ (C_{200}) for deciding when the individual and social learning phases occur as C_0 only has a 54% level of superiority over C_{200} (Table XVII), and the results show that Blondie24-RR has a 98% level of superiority over C_{200} (Table XX). It is worth mentioning that as C_{200} has very few epochs of social learning, the performance results should be very similar to C_0, which they are. This observation suggests that the uncertainty in performance due to a single run is small.

Based on the results in Tables XV–XX, it is not sensible to use $M = 50$ and $N = 100$ (C_{100}) or $M = 20$ and $N = 50$ (C_{50}) for deciding when individual learning and social learning should occur. Although C_{100} has a 68% level of superiority over C_0 (Table XVII) and C_{50} has a 97% level of superiority over C_0 (Table XVII), Blondie24-RR, which is a result of a simple modification to Blondie24, has a 95% level of superiority over C_{100} (Table XX) and has a 87% level of superiority over C_{50} (Table XX), so it is not worth using $M = 50$ and $N = 20$ (C_{20}) or $M = 100$ and $N = 50$ (C_{20}).

Based on the results obtained from Tables XVII and XX, it is clear that increasing the number of players in the social pool

TABLE XXI
SUMMARY OF WINS/LOSES WHEN NOT USING TWO MOVE BALLOT

		C_0	Blondie24-RR
C_1	Red	Lost	Lost
	White	Drawn	Lost
C_{200}	Red	Drawn	Lost
	White	Drawn	Lost
C_{100}	Red	Won	Lost
	White	Drawn	Lost
C_{50}	Red	Won	Lost
	White	Won	Drawn
C_{20}	Red	Won	Drawn
	White	Won	Drawn
C_{10}	Red	Won	Won
	White	Won	Won

will increase the performance of the algorithm. The results also show (see Table XX) that C_{10} shows a better performance than both C_0 and Blondie24-RR. Therefore, it is recommended to use $M = 5$ and $N = 10$ (C_{10}) to determine when the individual and social learning phases occur.

V. CONCLUSION AND FUTURE WORK

This paper has introduced an individual and social learning algorithm to an evolutionary *Checkers* program that is based on the Blondie24 architecture. The proposed algorithm shows promising results when tested against an implementation of an evolutionary *Checkers* program, called C_0, and also against a player obtained as a result of the authors' previous efforts to introduce a round robin tournament into Blondie24, called Blondie24-RR. For the purpose of the statistical test (using Bayeselo rating), and in order to make sure that C_0 and Blondie24-RR are not flukes of evolution, ten different C_0 and ten different Blondie24-RR players were evolved.

Six players were evolved in order to see the effects of increasing the number of players in the social pool. Each player was evolved using a selected pair of values for the individual and social learning phases.

We can also conclude that increasing the number of players in the social pool will increase the performance of the player. Therefore, it seems that the value of $M = 5$ is the best value to be used to determine where the individual learning phase occurs. The value of $N = 10$ was found to be the best value to be used to decide where the social learning occurs. In this case, C_{10} is the best player found. The results in Table XX show that C_{10} is better than Blondie24-RR. Based on the results, it would seem appropriate to use individual learning and social learning to enhance the evolutionary *Checkers* systems.

Our future work will investigate whether other changes are possible. For example, we will investigate the use of individual learning and social learning for solving the problem of Blondie24 being an *end product*. This will bring about a continuous learning paradigm. One possible way to do this is by keeping the best player from each generation in the social pool and, in the case of an evolved player losing a match against a stronger player (human or computer), then we either use the second best player from the pool or randomly select one and test it (if possible) against the same player. This procedure will continue to run every time the evolved player loses against human or computer players. This will not guarantee a win for the evolved player, but at least we will have a player that is able to continuously change its strategy when losing, in the hope of a win.

REFERENCES

[1] A. M. Turing, "Computing machinery and intelligence," *Mind*, vol. 59, pp. 433–460, 1950.

[2] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Develop.*, vol. 3, no. 3, pp. 210–229, 1959.

[3] D. B. Fogel, *Blondie24 Playing at the Edge of AI*. New York: Academic, 2002.

[4] K. Chellapilla and D. B. Fogel, "Evolving an expert checkers playing program without using human expertise," *IEEE Trans. Evol. Comput.*, vol. 5, no. 4, pp. 422–428, Aug. 2001.

[5] A. L. Samuel, "Some studies in machine learning using the game of checkers II—Recent progress," *IBM J. Res. Develop.*, vol. 11, no. 6, pp. 601–617, 1967.

[6] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[7] Mitchell and M. Tom, *Machine Learning*. New York: McGraw-Hill, 1997.

[8] R. S. Sutton and A. G. Barto, *Reinforcement Learning*. Cambridge, MA: MIT Press, 1998.

[9] J. Schaeffer, R. Lake, P. Lu, and M. Bryant, "Chinook: The world man-machine checkers champion," *AI Mag.*, vol. 17, no. 1, pp. 21–30, 1996.

[10] J. Schaeffer, *One Jump Ahead: Computer Perfection at Checkers*. New York: Springer-Verlag, 2009.

[11] J. Schaeffer, C. Joseph, and T. A. Norman, "World championship caliber checkers program," *Artif. Intell.*, vol. 53, no. 2–3, pp. 273–290, 1992.

[12] J. Schaeffer, N. Treloar, P. Lu, and R. Lake, "Man versus machine for the world checkers championship," *AI Mag.*, vol. 14, no. 2, pp. 28–35, 1993.

[13] J. Schaeffer, N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen, "Checkers is solved," *Sci. Exp.*, vol. 317, pp. 1518–1522, 2007.

[14] K. Chellapilla and D. B. Fogel, "Anaconda defeats Hoyle 6-0: A case study competing an evolved checkers program against commercially available software," in *Proc. Congr. Evol. Comput.*, La Jolla, CA, 2000, pp. 857–863.

[15] D. B. Fogel and K. Chellapilla, "Verifying anaconda's expert rating by competing against Chinook: Experiments in co-evolving a neural checkers player," *Neurocomputing*, vol. 42, pp. 69–86, 2002.

[16] K. Chellapilla and D. B. Fogel, "Evolution, neural networks, games, and intelligence," *Proc. IEEE*, vol. 87, no. 9, pp. 1471–1496, Sep. 1999.

[17] K. Chellapilla and D. B. Fogel, "Evolving neural networks to play checkers without relying on expert knowledge," *IEEE Trans. Neural Netw.*, vol. 10, no. 6, pp. 1382–1391, Nov. 1999.

[18] E. Harley, "Book Review: Blondie 24, playing at the edge of AI," *IEEE Comput. Intell. Bull.*, vol. 1, pp. 25–27, 2002.

[19] G. Kendall and Y. Su, "Imperfect evolutionary systems," *IEEE Trans. Evol. Comput.*, vol. 11, no. 3, pp. 294–307, Jun. 2007.

[20] B. Al-Khateeb and G. Kendall, "Introducing a round robin tournament into Blondie24," in *Proc. IEEE Symp. Comput. Intell. Games*, Milan, Italy, 2009, pp. 112–116.

[21] A. E. Elo, *The Rating of Chess Players, Past & Present*. New York: Arco, 1978.

[22] C. R. Bayeselo, "Bayesian Elo rating," 2005 [Online]. Available: http://remi.coulom.free.fr/Bayesian-Elo/
[23] D. R. Hunter, "MM algorithms for generalized bradley-terry models," *Ann. Stat.*, vol. 32, no. 1, pp. 384–406, 2004.
[24] H. A. Simon, *Administrative Behavior*, 4th ed. New York: The Free Press, 1997.
[25] C. D. Rosin and R. K. Belew, "New methods for competitive coevolution," *Evol. Comput.*, vol. 5, no. 1, pp. 1–29, 1997.
[26] G. Kendall and Y. Su, "The co-evolution of trading strategies in a multiagent based simulated stock market through the integration of individual and social learning," in *Proc. IEEE Congr. Evol. Comput.*, 2003, pp. 2298–2305.
[27] C. Chen, "Social learning mechanism in Java-Swarm artificial stock market," MSc thesis, Dept. Inf. Manage., National Center Univ., Taoyuan, Taiwan, 2004.
[28] R. Yamamoto, "Evolution with individual and social learning in an agent-based stock market," Computing in Economics and Finance Society for Computational Economics, No 228, 2005.
[29] S. Chen and C. Yeh, "Evolving traders and the business school with genetic programming: A new architecture of the agent-based artificial stock market," *J. Econ. Dyn. Control*, vol. 25, pp. 363–393, 2001.
[30] N. Vriend, "An illustration of the essential difference between individual and social learning, and its consequences for computational analysis," *J. Econ. Dyn. Control*, vol. 24, pp. 1–19, 2000.
[31] C. Spieth, F. Streichert, N. Speer, and A. Zell, "Utilizing an island model for EA to preserve solution diversity for inferring gene regulatory networks," in *Proc. Congr. Evol. Comput.*, 2004, vol. 1, pp. 146–151.
[32] S. Louis and C. Miles, "Playing to learn: Case-injected genetic algorithms for learning to play computer games," *IEEE Trans. Evol. Comput.*, vol. 9, no. 6, pp. 669–681, Dec. 2005.
[33] C. Miles, S. Louis, N. Cole, and J. McDonnell, "Learning to play like a human: Case injected genetic algorithms for strategic computer gaming," in *Proc. Congr. Evol. Comput.*, 2004, vol. 2, pp. 1441–1448.
[34] R. G. Reynolds, "An adaptive computer model of the evolution of agriculture for hunter-gatherers in the Valley of Oaxaca," Ph.D. dissertation, Dept. Comput. Sci., Univ. Michigan, Ann Arbor, MI, 1979.
[35] R. G. Reynolds, "An introduction to cultural algorithms," in *Proc. 3rd Annu. Conf. Evol. Program.*, 1994, pp. 131–139.
[36] M. Sternberg and R. G. Reynolds, "Using cultural algorithms to support re-engineering of rule-based expert systems in dynamic environments: A case study in fraud detection," *IEEE Trans. Evol. Comput.*, vol. 1, no. 4, pp. 225–243, Nov. 1997.
[37] R. G. Reynolds, Z. Kobti, and T. A. Kohler, "The effects of generalized reciprocal exchange on the resilience of social networks: An example from the prehistoric Mesa Verde region," *J. Comput. Math. Organ. Theory*, vol. 9, no. 3, pp. 229–254, 2003.
[38] R. G. Reynolds, Z. Kobti, T. A. Kohler, and L. Yap, "Unravelling ancient mysteries: Reimagining the past using evolutionary computation in a complex gaming environment," *IEEE Trans. Evol. Comput.*, vol. 9, no. 6, pp. 707–720, Dec. 2005.
[39] R. G. Reynolds and S. Saleem, "Function optimization with cultural algorithms in dynamic environments," in *Proc. IEEE Particle Swarm Optim. Workshop*, 2001, pp. 63–79.
[40] R. G. Reynolds and B. Peng, "Cultural algorithms: Knowledge learning in dynamic environments," *Proc. IEEE Int. Congr. Evol. Comput.*, pp. 1751–1758, 2004.
[41] R. G. Reynolds and S. Saleem, "The impact of environmental dynamics on cultural emergence," in *Perspectives on Adaptation in Natural and Artificial Systems-Essays in Honor of John Holland*, L. Booker, S. Forrest, M. Mitchell, and R. Riolo, Eds. London, U.K.: Oxford Univ. Press, 2004, pp. 253–280.
[42] E. A. Feigenbaum and J. Feldman, Eds., *Computer and Thought*. New York: McGraw-Hill, 1963.
[43] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Develop.*, pp. 207–226, 2000.
[44] D. L. Levy, Ed., *Computer Games*. New York: Springer-Verlag, 1988, pp. 366–400.

Belal Al-Khateeb received the B.Sc. (honors; first class) and M.Sc. degrees in computer science from Al-Nahrain University, Baghdad, Iraq, in 2000 and 2003, respectively, and the Ph.D. degree from the School of Computer Science, University of Nottingham, Nottingham, U.K., in 2011.

Currently, he is a Lecturer at the College of Computer, Al-Anbar University, Ramadi, Iraq. He has published over 14 refereed journal and conference papers. His current research interests include evolutionary and adaptive learning, particularly in computer games, expert systems, and heuristics and meta/hyperheuristics. He has a particular interest in computer games programming.

Dr. Al-Khateeb is a reviewer of two international journals (including one IEEE TRANSACTIONS) and four conferences.

Graham Kendall (M'03–SM'10) received the B.S. (honors; first class) degree in computation from the University of Manchester Institute of Science and Technology (UMIST), Manchester, U.K., in 1997 and the Ph.D. degree from the School of Computer Science, University of Nottingham, Nottingham, U.K., in 2000.

Before entering academia, he spent almost 20 years in the IT industry, working for various U.K. companies, taking on roles such as Computer Operator, Technical Support Manager, and Service Manager. He is currently the Dunford Professor of Computer Science with the Automated Scheduling, Optimization, and Planning Research Group, School of Computer Science, University of Nottingham. He also holds the position of Vice-Provost (Research and Knowledge Transfer) at Nottingham's Malaysia Campus. He has edited and authored nine books and has published over 175 refereed journal and conference papers. His current research interests include scheduling, optimization, evolutionary and adaptive learning, and heuristics and meta/hyperheuristics. He has a particular interest in solving real-world problems.

Prof. Kendall is a Fellow of the Operational Research Society and an Associate Editor of seven international journals (including two IEEE TRANSACTIONS). He was one of the cofounders of the IEEE Symposium of Computational Intelligence in Games and he also chairs (since 2003) the Steering Committee of the Multidisciplinary International Conference on Scheduling: Theory and Applications.