A Self-Learning Evolutionary Chess Program

DAVID B. FOGEL, FELLOW, IEEE, TIMOTHY J. HAYS, SARAH L. HAHN, AND JAMES QUON

Contributed Paper

A central challenge of artificial intelligence is to create machines that can learn from their own experience and perform at the level of human experts. Using an evolutionary algorithm, a computer program has learned to play chess by playing games against itself. The program learned to evaluate chessboard configurations by using the positions of pieces, material and positional values, and neural networks to assess specific sections of the chessboard. During evolution, the program improved its play by almost 400 rating points. Testing under simulated tournament conditions against Pocket Fritz 2.0 indicated that the evolved program performs above the master level.

Keywords*—Chess, computational intelligence, evolutionary computation, machine learning, neural networks.*

I. INTRODUCTION

A fundamental approach to artificial intelligence has been to capture human expertise in the form of programmed rules of behavior that, when executed on a computer, emulate the behavior of the human expert. Simulating the symptoms of intelligent behavior rather than the mechanisms that underlie that behavior was a natural outgrowth of the pressure to design useful software applications. This approach, however, presents a significant limitation: artificial intelligence programs are restricted mainly to those problems that are, in large part, already solved. The challenge of creating machines that can learn from their own experience without significant human intervention has remained elusive.

Efforts in machine learning have often been focused on reinforcement learning, in which a series of actions leads to success or failure and some form of credit or blame for the final result is apportioned back to each action [1]–[3]. In a game of strategy, such as chess, the final result of win, lose, or draw is associated with numerous decisions made from the first to the final move. The *credit assignment problem* then has been to find rules for rewarding "correct" actions and penalizing "incorrect" actions. It must be acknowledged, however, that describing any particular action as correct or incorrect may be vacuous because the end result is typically a nonlinear function of the interaction of the moves played by both players. The final result is more than simply the sum of the attributed worth of each move in a series. Nevertheless, the credit assignment approach has been adopted largely based on a long-standing belief that there is insufficient information in "win, lose, or draw" when referred to the entire game to "provide any feedback for learning at all over available time scales."[1]

Experiments described here indicate that in contrast, a machine learning algorithm based on the principles of Darwinian evolution [4] can adapt a chess program that plays at a rating that is commensurate with a grandmaster without relying on any specific credit assignment algorithms.[2] Not only is the outcome of any specific game not used as direct feedback, only a point score reflecting performance over a series of games is used to judge the performance of competing programs. This protocol is adopted to indicate the versatility of the evolutionary approach.

II. BACKGROUND

For more than 50 years, the game of chess has served as a testing ground for efforts in artificial intelligence, both in terms of computers playing against other computers and computers playing against humans [5]–[12]. During these five decades, the progress of chess programs in terms of their measured performance ratings has been steady. This progress, however, has not arisen mainly because of any real improvements in anything that might be described as "artificial intelligence."

Instead, progress has come most directly from the increase in the speed of computer hardware [13], and also straightforward software optimization. *Deep Blue*, the famous computer program and hardware (32 computing nodes, each with eight very-large-scale integrated processors) that defeated Kasparov in 1997, evaluated 200 million alternative

Manuscript received June 15, 2004; revised September 7, 2004. This work was sponsored in part by the National Science Foundation under Grants DMI-0232124 and DMI-0349604.

The authors are with Natural Selection, Inc., La Jolla, CA 92037 USA (e-mail: dfogel@natural-selection.com).

Digital Object Identifier 10.1109/JPROC.2004.837633

[1]A. Newell, quoted in [2].

[2]Earning an official designation of grandmaster requires competition in sanctioned tournaments against other grandmasters.

positions per second. By contrast, the computer that executed *Belle*, the first program to earn the title of U.S. master in 1983, was more than 100 000 times slower. Faster computing and optimized programming allows a chess program to evaluate chessboard positions further into the prospective future. Such a program can then choose current moves that can be expected to lead to better outcomes that might not be seen by a program running on a slower computer or with inefficient programming.

Chess programs rely typically on a database of opening moves and endgame positions and use a mathematical function to evaluate intermediate positions. This function usually comprises features regarding the values assigned to individual pieces (material strength), mobility, tempo, and king safety, as well as tables that are used to assign values to pieces based on their position (positional values) on the chessboard. The parameters that govern these features are set by human experts, but can be improved upon by using an evolutionary algorithm. Furthermore, an evolutionary algorithm can be employed to discover features that lead to improved play.

To accomplish this, the evolutionary program described here used three artificial neural networks (i.e., networked layers of nonlinear processing elements) that were used to evaluate the worth of alternative potential positions in sections of the chessboard (front, back, and middle). The procedure started with a population of alternative simulated players, each initialized to rely on standard material and positional values taken from open-source chess programs, and supplemented each player with the three neural networks. The simulated players then competed against each other for survival and the right to generate "offspring" through a process of random variation applied to the standard parameters and neural connection weights.

Survival was determined by the quality of play in a series of chess games played against opponents from the same population. Over successive generations of variation and selection, the surviving players extracted information from the game and improved their performance. The best-evolved player was tested against *Chessmaster 8000*, a popular commercial program, in 120 games and attained a rating that was almost 400 rating points higher than the open-source programs used to start the evolutionary learning process. The evolved player was then tested in simulated tournament conditions in 12 games (six as black, six as white) against *Pocket Fritz 2.0*, a standard chess program that plays at a rating of 2300–2350 (high-level master). The evolved player won this contest with nine wins, two losses, and one draw.

III. Method

The rules for chess are well known and are found easily on the Internet.[3] A chess engine was provided by Digenetics, Inc., La Jolla, CA, and extended for the current experiments. The baseline chess program functioned as follows. Each chessboard position was represented by a vector of length 64, with each component in the vector corresponding to an available position on the board. Components in the vector could take on values from $\{-K, -Q, -R, -B, -N, -P, 0, +P, +N, +B, +R, +Q, +K\}$, where 0 represented an empty square and the variables P, N, B, R, Q, and K represented material values for pawns, knights, bishops, rooks, and the queen and king, respectively. The chess engine assigned a material value to kings even though the king cannot actually be captured during a match. The sign of the value indicated whether or not the piece in question belonged to the player (positive) or the opponent (negative).

A player's move was determined by evaluating the presumed quality of potential future positions. An evaluation function was structured as a linear combination of: 1) the sum of the material values attributed to each player; 2) values derived from tables that indicated the worth of having specific pieces in specific locations on the board, termed "positional value tables" (PVTs); and 3) three neural networks, each associated with specific areas of the chessboard. Each piece type other than a king had a corresponding PVT that assigned a real value to each of the 64 squares, which indicated the presumptive value of having a piece of that type in that position on the chessboard. For kings, each had three PVTs: one for the case before a king had castled, and the others for the cases of the king having already castled on the kingside or queenside. The PVTs for the opponent were the mirror image of the player's PVTs (i.e., rotated 180°). The entries in the PVTs could be positive and negative, thereby encouraging and discouraging the player from moving pieces to selected positions on the chessboard. The nominal (i.e., not considering the inclusion of neural networks) final evaluation of any position was the sum of all material values plus the values taken from the PVTs for each of the player's own pieces (as well as typically minor contributions from other tables that were used to assess piece mobility and king safety for both sides). The opponent's values from the PVTs were not used in evaluating the quality of any prospective position.

Games were played using an alpha–beta search [14] of the associated game tree for each board position looking a selected number of moves into the future. With the exception of moves made from opening and endgame databases, the minimax move for a given ply (two ply corresponds to a move from each player) was determined by selecting the available move that afforded the opponent the opportunity to do the least damage as determined by the evaluation function on the resulting position. The depth of the search was set to four ply to allow for reasonable execution times in the evolutionary computing experiments (50 generations on a 2.2-GHz Celeron with 128 MB of RAM required 36 h). Fig. 1 shows the flowchart for a typical evolutionary algorithm, and Fig. 2 shows the architecture for a simple neural network. The search depth was extended in particular situations as determined by a quiescence routine that checked for any possible piece captures, putting a king in check, and passed pawns that had reached at least the sixth rank on the board (anticipating pawn promotion), in which case the ply depth was extended by two (following earlier work [15], [16]

[3]http://www.uschess.org/beginners/letsplay.php

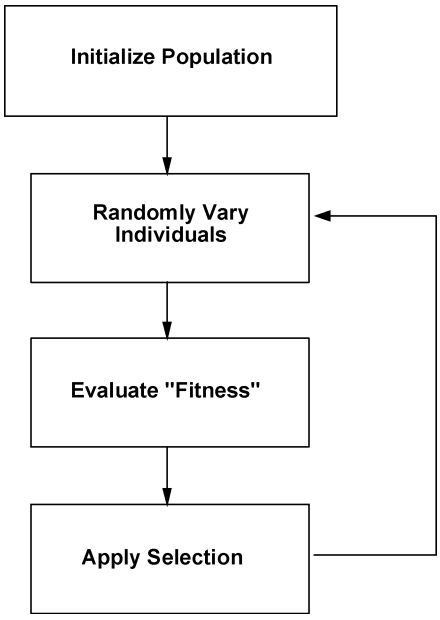

Fig. 1. Flowchart for the evolutionary algorithm. A population of candidate chess-playing programs is created, perhaps completely at random. These "parent" players are used to create offspring by randomly varying their components. The worth (fitness) of each parent and offspring is evaluated in head-to-head competition. Selection is applied to eliminate those players that do not score well relative to others in the population. The survivors become the parents for the next iteration (generation).

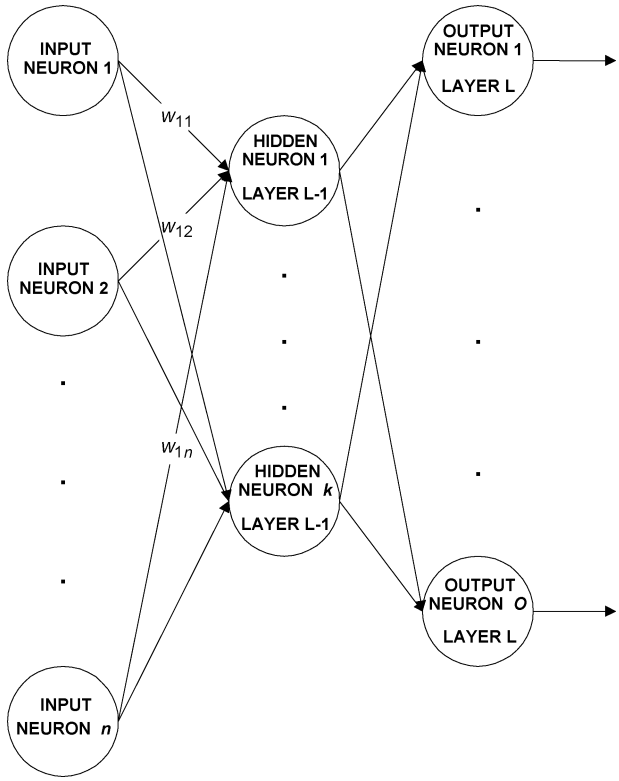

Fig. 2. Typical neural network architecture. Artificial neurons are connected via variable weights. The input neurons sense objects in a scene (such as chess pieces on a board represented by numeric values). The dot product of the weights and input neuron values (known as activations) is computed at each "hidden neuron" and passed through a nonlinear transformation, typically a sigmoid function. The outputs of the hidden neurons are then processed similarly until reaching the output neurons, which compute the observable output of the neural network. For the case of evolving neural networks in chessboard evaluation, there was only one output neuron, representing the imputed worth of the position of the pieces in the sensed area of the chessboard. Object neural networks focus on specific objects or regions in a scene.

that evolved strategies for the game of checkers). The best move to make was chosen by iteratively minimizing or maximizing over the leaves of the game tree at each ply according to whether or not that ply corresponded to the opponent's move or the player's. The games were executed until one player suffered checkmate, upon which the victor was assigned a win and the loser was assigned a loss, or until a position was obtained that was a known draw (e.g., one king versus one king) or the same position was obtained three times in one game (i.e., a threefold repetition draw), or if 50 total moves were exceeded for both players. (This should not be confused with the so-called 50-move rule for declaring a draw in competitive play.) Points were accumulated, with players receiving $+1$ for a win, 0 for a draw, and -1 for a loss.

A. Initialization

The evolutionary experiment was initialized with a population of 20 computer players (ten parents and ten offspring in subsequent generations) each having nominal material values and entries in their PVTs and randomized neural networks.

The initial material values for P, N, B, R, Q, and K were 1, 3, 3, 5, 9, and 10 000, respectively. The king value was not mutable.

The initial entries in the PVTs were in the range of -50 to $+40$ for kings, -40 to $+80$ for queens and rooks, -10 to $+30$ for bishops and knights, and -3 to $+5$ for pawns, and followed values gleaned from other open-source chess programs.

Three object neural networks were added to the protocol described in the baseline experiment. The neural networks were fully connected feedforward networks with 16 inputs, ten hidden nodes, and a single output node. The neural networks focused on the first two rows, the back two rows, and the center of the chessboard, respectively (Fig. 3). These choices were made to reflect areas of known importance for protecting one's own territory, advancing into the opponent's territory, and control of the center. The choice of ten hidden nodes was arbitrary, and future efforts will be made to adjust this structure and allow the neural networks' topologies to evolve along with their weights and connections. The hidden nodes used standard sigmoid transfer functions $f(x) = 1/(1 + \exp(-x))$, where x was the dot product of the incoming activations from the chessboard and the associated weights between the input and hidden nodes, offset by each hidden node's bias term. The output nodes also used the standard sigmoid function but were scaled in the range of $[-50, 50]$, on par with elements of the PVTs. The outputs of the three neural networks were added to the material and PVT values to come to an overall assessment of each alternative board position. All weights and biases

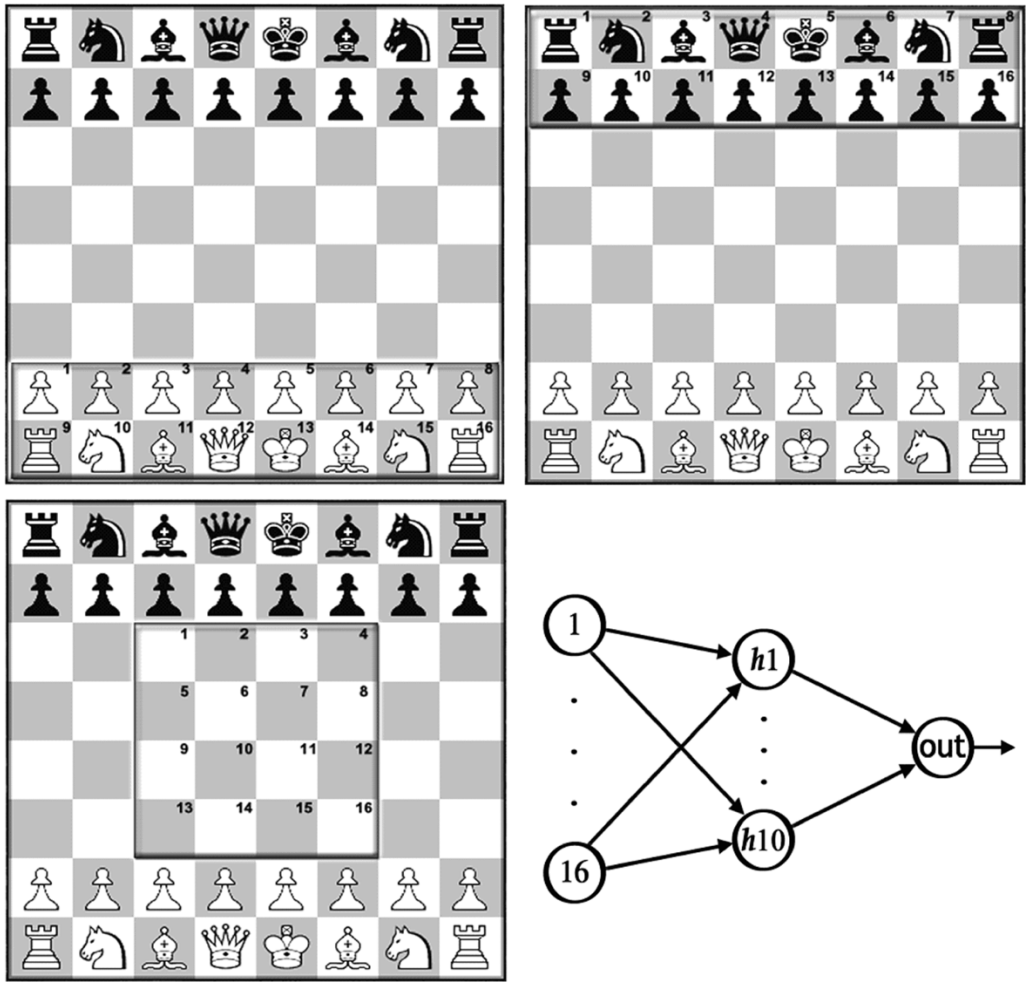

Fig. 3. Three chessboards indicate the areas in which the object neural networks focused attention. The upper-left panel highlights the player's front two rows. The 16 squares as numbered were used for inputs to a neural network. The upper-right panel highlights the back two rows, and the contents were similarly used as input for a neural network. The lower-left panel highlights the center of the chessboard, which was again used as input for a neural network. In all, there were three neural networks, one for each focused section. Each network was designed as shown in the lower-right panel, with 16 inputs (as numbered for each of the sections), ten hidden nodes (h1–h10), and a single output node. The bias terms on the hidden and output are not shown.

were initially distributed randomly in accordance with a uniform random variable $U(-0.025, 0.025)$ and initial strategy parameters were distributed $U(0.05)$.

Candidate strategies for the game were, thus, represented in the population as the material values, the PVT values, the weights and bias terms of the three neural networks, and associated self-adaptive strategy parameters for each of these parameters, explained as follows.

B. Variation

One offspring was created from each surviving parent by mutating all (each one of) the parental material, PVT values, and weights and biases of all three neural networks. Mutation was implemented on material values according to

$$m_i' = m_i + N(0, s_i') \tag{1}$$

where $N(\mu, \sigma)$ is a sample from a Gaussian random variable with mean μ and standard deviation σ, m_i' is the material value of the ith element assigned to the offspring, m_i is the material value of the piece in question (ith index position) for the parent, and s_i' is a self-adaptive step size (variance), called a strategy parameter, for the ith index position that was evolved following the standard practice of lognormal perturbations

$$s_i' = s_i \times \tau \times \exp(N(0, 1)) \tag{2}$$

where $\tau = 1/\mathrm{sqrt}(2n)$, and n was the number of evolvable parameters, which varied in each experimental setup. The material value and PVT strategy parameters were set initially to random samples from $U(0{,}0.05)$, where U is the uniform distribution. Mutation for the PVT values, as well as each of the weights and biases of the neural networks, followed the same Gaussian form of mutation as applied to material values. In the case where a mutated material value took on a negative number, it was reset to zero.

Table 1
Results of Playing the Best-Evolved Player From Each of the Ten Trials With Evolvable Material, PVT, and Three Feedforward Fully Connected Object Neural Networks as Black Against the Nonevolved Player in 200 Games

Trial	Wins	Losses	Draws	Win%	*Z*-score
1	65	55	80	0.542	0.920
2	72	51	77	0.585	1.886
3	58	30	112	0.659	2.983
4	66	53	81	0.555	1.200
5	73	55	72	0.570	1.584
6	80	55	65	0.593	2.161
7	62	29	109	0.681	3.453
8	126	45	29	0.737	6.198
9	81	47	72	0.633	3.009
10	82	58	60	0.586	2.035

The results show statistically significant evidence that the evolved player is superior to the nonevolved player (sign-test, $P < 0.05$). Win% indicates the ratio of wins to decisions. Z-score indicates the z-statistic for a probability test (null hypothesis of $p_0 = 0.5$) and 6 of 10 trials result in data that are statistically significant ($\alpha = 0.05$, $z^* = \pm 1.96$; Note that a one-sided test could be adopted but the more conservative two-sided test is reported here).

C. Selection

Competition for survival was conducted by having each player play ten games (five as white and five as black) against randomly selected opponents from the population (with replacement, not including itself). The outcome of each game was recorded and points summed for all players in all games. After all 20 players completed their games, the ten best players according to their accumulated point totals were retained to become parents of the next generation.

D. Experimental Design

A series of ten independent trials was conducted. The available computational capability (stated earlier) limited the population size to ten parents and ten offspring in each trial. Fifty generations were executed in each of ten trials, whereupon the best-evolved player was tested as black against the initial nonevolved chess engine in 200 games. The results of these competitions are indicated in Table 1. All ten trials favored the evolved player over the nonevolved player (sign-test favoring the evolved player, $P < 0.05$), indicating a replicable result.

IV. Results

The best result was observed in trial 8, which evidenced the greatest number of wins against the nonevolved player and also the greatest difference in wins and losses. (The corresponding evolved player from this trial was named *Blondie25*, following earlier research using a similar protocol that evolved a checkers-playing program called *Blondie24* [13], [17]). The evolved program's performance was assessed at six-ply by playing 120 games (60 as black) against the commercial program *Chessmaster 8000*, which is a popular commercial game that has over a decade of history, published in several successively updated versions. *Chessmaster 8000* offers computer opponents that can play at indicated rating levels. The chess rating system is the one adopted by the United States Chess Federation (USCF), which uses the formula

$$R_{\text{New}} = R_{\text{Old}} + C(\text{Outcome} - W) \tag{3}$$

where $W = 1/(1+10\text{^}[(R_{\text{Opp}} - R_{\text{Old}})/400])$, Outcome is 1 if a win, 0.5 if a draw, and 0 if a loss, R_{Opp} is the opponent's rating, and $C = 32$ for ratings less than 2100, $C = 24$ for ratings between 2101 and 2400, and $C = 16$ for ratings above 2400. The rating system awards more points to players who defeat opponents that are rated higher, and takes away more points from players who lose to presumed lesser competition. The standard class intervals are designated in Table 2.

Fig. 4 shows the histogram of the results separated by intervals of 100 rating points. An iterative statistical procedure for determining the evolved player's rating was employed and yielded a final estimate of 2437 (± 3.19) after 5000 perturbations of the ordering of the 120 games played. The best results were a win against *Chessmaster 8000* playing at an indicated rating of 2818, and a draw at 2860. The win over *Chessmaster 8000* at 2818 came after *Chessmaster 8000* was leading by two pawns and blundered (perhaps as the result of a software bug or logical flaw) and does not represent performance at the level of world-class players.

To quantify a baseline control measure to calibrate the increase in performance that the evolutionary procedure provided, an additional 120 games were played against *Chessmaster 8000* with the nonevolved program, absent the inclusion of the neural networks (i.e., relying on the standard facets of the chess program). Fig. 5 shows the histogram of

Table 2
Relevant Categories of Player Indicated By the Corresponding Range of Rating Score

Class	Rating
Senior Master	2400+
Master	2200-2399
Expert	2000-2199
Class A	1800-1999
Class B	1600-1799
Class C	1400-1599
Class D	1200-1399
Class E	1000-1199
Class F	800-999
Class G	600-799
Class H	400-599
Class I	200-399
Class J	Below 200

The title of "Senior Master" is used by the USCF. FIDE, an international chess organization, confers titles of FIDE Master, International Master, and Grandmaster based on rating and tournament performance.

results. The nonevolved program's rating was estimated at 2066 ($\pm$3.56), indicating that the evolutionary procedure accounted for an increase in 371 rating points over the nonevolved player.

Many chess devotees spend considerable effort in determining and publishing the rating of chess programs. One such listing[4] indicates that the current best chess program is *Shredder 7.04*, tested on an Athlon 1200 MHz with 256 MB of RAM, with a rating of 2808 $\pm$ 26 across over 866 tested games. By contrast, *Chessmaster 8000* rates at 2517 (Athlon 450 MHz/128 MB), and the closely related *Chessmaster 9000* rates at 2718 (Athlon 1200 MHz/256 MB), both lower than the indicated 2800+ rating offered when executing the *Chessmaster 8000* program in the above experiments. Thus the rating of 2437 for the evolved player at only six ply cannot be used reliably for claiming any absolute level of performance.

At the suggestion of Kasparov [18] the best-evolved program was tested (using an Athlon 2400+/256 MB) against *Pocket Fritz 2.0* under simulated tournament conditions, which provide 120 min for the first 40 moves, 60 min for the next 20 moves, and an additional 30 min for all remaining moves. Unlike *Pocket Fritz 2.0* and other standard chess programs, the evolved player does not treat the time per move dynamically. The time per move was prorated evenly across the first 40 moves after leaving the opening book, with 3 min per move allocated to subsequent moves. *Pocket Fritz 2.0* was executed on a "pocket PC" running at 206 MHz/64 MB RAM, with all computational options set to their maximum strength. *Pocket Fritz 2.0* is known to play at the level of 2300–2350 (high-level master) under these conditions [18].

A series of 12 games were played, with the evolved program playing six as black and six as white. The evolved program won nine, lost two, and drew one. The move lists for each of these games are available[5] along with annotation by one of the authors of this paper (J. Quon), a national chess master who is rated 2301. The results provide evidence to estimate a so-called "performance rating" of the evolved player under tournament settings at approximately 2550, about 250 points higher than *Pocket Fritz 2.0*, and is to our knowledge the first result of a program that has taught itself to play chess at this high level of performance.

V. Discussion

The best-evolved chess player cannot yet compete with other chess programs at the highest levels of play (e.g., *Shredder*, *Deep Fritz*, *Deep Junior*). Yet the evolutionary program exhibits a flexibility that cannot be achieved with these programs, each of which relies on its "intelligence" being preprogrammed. The evolutionary algorithm is capable of adapting its play to meet more demanding challenges from superior opponents. It can invent new unorthodox tactics. Indeed, in earlier research that evolved strategies for playing checkers (also known as draughts) [13], it was common for human opponents to offer comments that the evolved player's moves were "strange," yet the level of play was often described as "very tough" or with other complimentary terms.

Although the evolution of a high-level chess program is itself significant, it signifies the broader utility of the evolutionary approach to problem solving. This approach to computational intelligence does not complete an exhaustive search of all possible designs, yet it can quickly identify and converge on useful solutions. This may provide an effective means for going beyond the structured conventional engineering approach that is common to many forms of human design. The process can be used to address problems for which there are no known solutions, or it can be hybridized with existing human knowledge and discover ways to supplement that knowledge. The evolutionary process is inherently parallel and could be accelerated greatly by utilizing a distributed network of small and inexpensive processors.[6] Furthermore, no explicit design criteria are required. At a minimum, all that is necessary is to be able to compare two solutions and determine which is better. The process of evolution can then optimize the solution for the given problem. In this manner, evolutionary machine learning can meet, supplement, and potentially exceed the capabilities of human expertise.

[4]The Swedish Chess Computer Association publishes ratings of the top 50 computer programs at http://w1.859.telia.com/~u85 924 109/ssdf/list.htm

[5]The annotated moves are available at http://www.natural-selection.com/chessgames.html

[6]See http://neural-chess.netfirms.com/HTML/project.html

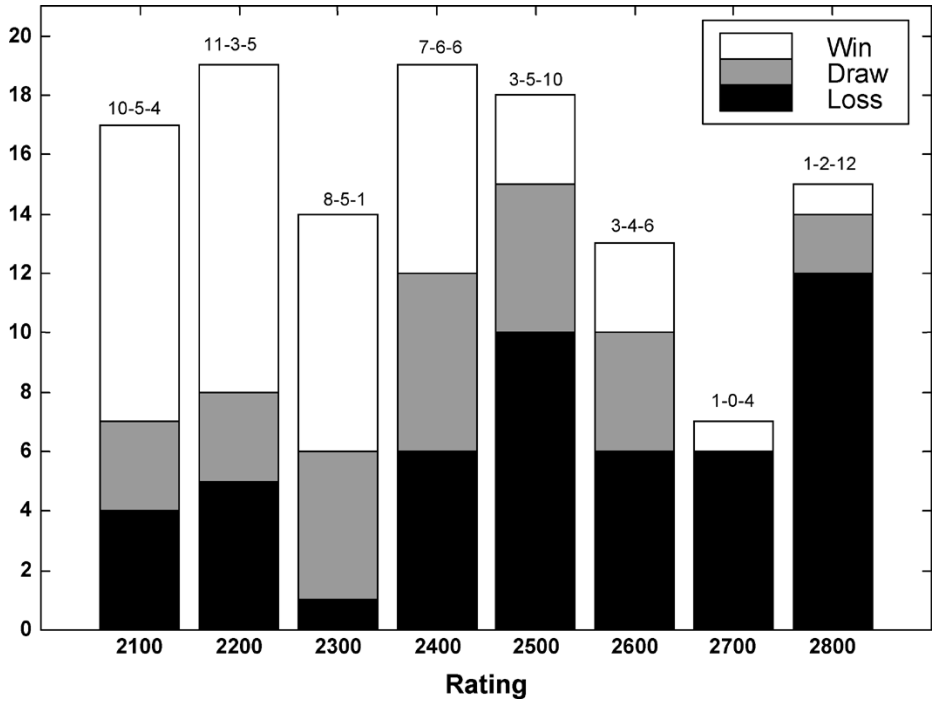

Fig. 4. Histogram of wins, draws, and losses for the best evolved player against *Chessmaster 8000*. Each rating category represents a 100-point range (e.g., 2100 represents 2100–2199).

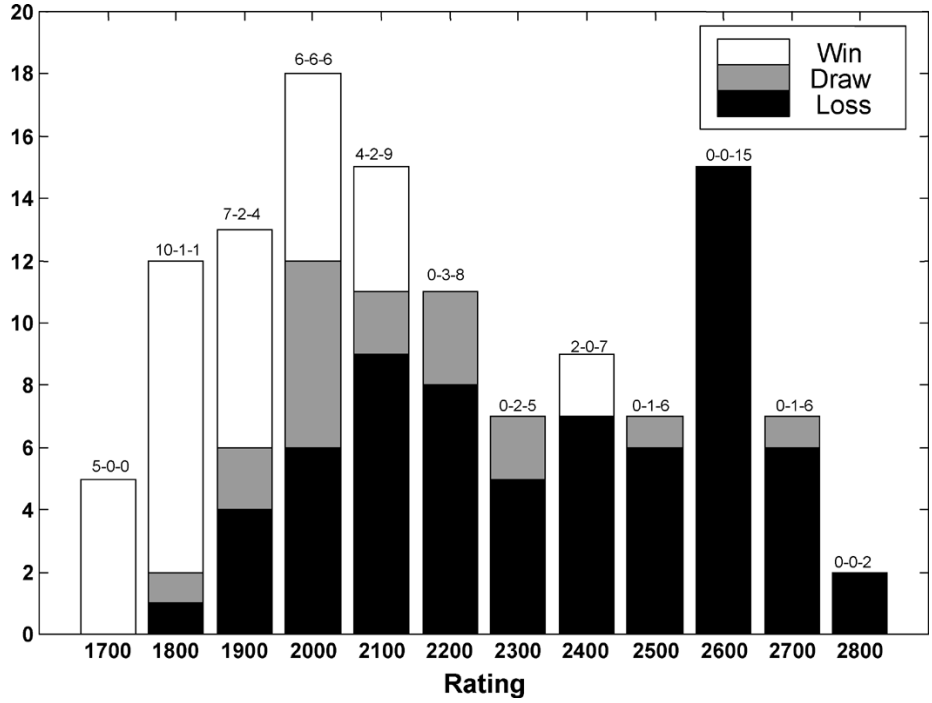

Fig. 5. Histogram of wins, draws, and losses for the nonevolved player against *Chessmaster 8000*. The results are much poorer than those of the best evolved player, which rated 379 points higher.

ACKNOWLEDGMENT

The authors would like to thank G. Kasparov for his valuable insights, W. Atmar, G. Fogel, and the anonymous reviewers for helpful criticisms, D. Johnson and Digenetics, Inc., for making their chess engine available, and G. Burgin for assistance with figures. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," *IBM J. Res. Dev.*, vol. 3, pp. 210–229, 1959.

[2] M. L. Minsky, "Steps toward artificial intelligence," *Proc. IRE*, vol. 49, pp. 8–30, 1961.
[3] G. Tesauro, "Practical issues in temporal difference learning," *Mach. Learn.*, vol. 8, pp. 257–277, 1992.
[4] D. B. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*, 2nd ed. New York: IEEE Press, 1999.
[5] C. E. Shannon, "Programming a computer for playing chess," *Philos. Mag.*, vol. 41, pp. 256–275, 1950.
[6] A. M. Turing, "Digital computers applied to games," in *Faster Than Thought*, B. V. Bowden, Ed. London, U.K.: Pittman, 1953, pp. 286–310.
[7] A. Newell, J. C. Shaw, and H. A. Simon, "Chess-playing programs and the problem of complexity," *IBM J. Res. Dev.*, vol. 2, pp. 320–325, 1958.
[8] D. N. L. Levy and M. Newborn, *How Computers Play Chess*. New York: Comput. Sci., 1991, pp. 28–29, 35–39.
[9] B. Cipra, "Will a computer checkmate a chess champion at last?," *Science*, vol. 271, p. 599, 1996.
[10] J. McCarthy, "AI as sport," *Science*, vol. 276, pp. 1518–1519, 1997.
[11] A. B. Markman, "If you build it, will it know?," *Science*, vol. 288, pp. 624–625, 2000.
[12] C. Holden, "Draw in Bahrain," *Science*, vol. 298, p. 959, 2002.
[13] D. B. Fogel, *Blondie24: Playing at the Edge of AI*. San Francisco, CA: Morgan Kaufmann, 2002.
[14] H. Kaindl, "Tree searching algorithms," in *Computers, Chess, and Cognition*, T. A. Marsland and J. Schaeffer, Eds. New York: Springer-Verlag, 1990, pp. 133–168.
[15] ——, "Evolution, neural networks, games, and intelligence," *Proc. IEEE*, vol. 87, pp. 1471–1496, Sept. 1999.
[16] K. Chellapilla and D. B. Fogel, "Evolving an expert checkers playing program without using human expertise," *IEEE Trans. Evol. Comput.*, vol. 5, pp. 422–428, Aug. 2001.
[17] I. Aleksander, "Neural networks: evolutionary checkers," *Nature*, vol. 402, pp. 857–860, 1999.
[18] G. Kasparov, private communication, 2004.

David B. Fogel (Fellow, IEEE) received the Ph.D. degree in engineering sciences from the University of California at San Diego, La Jolla, in 1992.

He was a Systems Analyst for Titan Systems, Inc. (1984–1988), and a Senior Principal Engineer at ORINCON Corporation (1988–1993). He is currently Chief Executive Officer of Natural Selection, Inc., La Jolla. His experience includes over 19 years of applying computational intelligence methods and statistical experimental design to real-world problems in industry, medicine, and defense. He has taught undergraduate and graduate courses in evolutionary computation, stochastic processes, and statistical process control. He has over 200 publications in the technical literature, the majority treating the science and application of evolutionary computation, and is the author or coauthor of six books. He also serves as the Editor-in-Chief of *BioSystems* and is on the editorial boards of other technical journals.

Dr. Fogel has received several honors and awards, including the 2002 Sigma Xi Southwest Region Young Investigator Award, the 2003 Sigma Xi San Diego Section Distinguished Scientist Award, the 2003 SPIE Computational Intelligence Pioneer Award, and most recently the 2004 IEEE Kiyo Tomiyasu Technical Field Award. He was the Founding Editor-in-Chief of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION (1996–2002). He was the Founding President of the Evolutionary Programming Society (1991–1993). He served as the General Chairman for the 2002 IEEE World Congress on Computational Intelligence and will serve as the General Chairman for the 2007 IEEE Computational Intelligence Symposium Series, April 1–5, in Honolulu, HI.

Timothy J. Hays has studied at seven colleges in the areas of computer science, VR design, physics, chemistry and biology, instrumental music, and electronics.

He has 27 years of software product development experience and pioneered programs available on floppy disks, embedded ROM systems, CDs, and DVDs. He has worked with Cinemaware, Sony, Sega, Virgin, Shiny, Interplay, and Electronic Arts and has completed over 70 published software products. He was the Lead Programmer on Evolutionary Checkers Starring Blondie24 (Digenetics, 2001) and Chess with an Attitude! (Digenetics, 2003). Focusing on multiplayer artificial intelligence (AI), physical-model/physics engines, and real-time three-dimensional (3-D) graphics simulations, he has created titles in many entertainment genres such as Fox Studios' The X-Files, Sony Gameday Football, Terry Bradshaw Football '97, Sony ESPN Baseball Tonight, Electronic Arts NHL Hockey '93 to '98, Virgin Games' Muhammad Ali Boxing, Shiny's Messiah, Shiny's R/C Stunt Copter, and a WWI flight simulator titled "Wings," which was the best-selling game of all time for Amiga computers (Cinemaware, 1990), MacAttack (first 3-D game available for Macintosh computers), and Harrier Strike Mission I and II (best-sellers on early Macintoshes), on consoles such as the Sony Playstation, Nintendo, Sega Genesis, DreamCast and Saturn, Bally Arcade, Atari series computers, Amigas, and Windows PCs. His knowledge of combining highly optimized assembly language with C/C++ code allowed him to create products with some of the fastest graphics (both 2-D and 3-D) available on several platforms. He has also created several business training (CBT) courses and three MBA degree courses, functioning as a Producer and Designer. He is currently a Senior Staff Scientist at Natural Selection, Inc., La Jolla, CA. He has extensive knowledge of video codecs/video compression, video editing, SFX, 3-D graphics, Photoshop, streaming video, and Flash animation and has used his knowledge and programming skills to create products for several companies using his own multimedia engine with high-quality video support on CD and DVD.

Sarah L. Hahn received the B.A. degree in English from Pomona College, Claremont, CA, in 2001.

She taught middle school language arts with Teach for America. She is currently Director of Communications for Natural Selection, Inc., La Jolla, CA. She is responsible primarily for maintaining effective corporate communication with both the scientific community and the greater public.

Ms. Hahn is a member of the Association for the Study of Literature and the Environment. She is also active in the community, volunteering as Cochair and an Outings Leader for a nonprofit group, San Diego Inner City Outings.

James Quon is currently working toward the degree in computer science, with a specialization in artificial intelligence, from the University of California at San Diego, La Jolla.

He is a Staff Scientist with Natural Selection, Inc., La Jolla, CA. He is also a nationally ranked chess master and professional chess coach. He has over 15 years of teaching experience, and his students have gone on to win tournaments at all levels. Since he moved to San Diego, his teams have won nine state championship titles. He also donates much of his time teaching chess to underprivileged children in the community who would not otherwise have the opportunity to take chess lessons. He has written for *Rank and File* magazine.

Mr. Quon has served on the Southern California Chess Federation board.