Playing Chess at a Human Desired Level and Style

ABSTRACT

Human chess players prefer training with human opponents over chess agents as the latter are distinctively different in level and style than humans. Chess agents designed for human-agent play are capable of adjusting their level, however their style is not aligned with that of human players. In this paper, we propose a novel approach for designing such agents by integrating the theory of chess players' decision-making with a state-of-the-art Monte Carlo Tree Search (MCTS) algorithm. We demonstrate the benefits of our approach using two sets of analyses. Quantitatively, we establish that the agents attain their desired Elo ratings. Qualitatively, through a Turing-inspired test with a human chess expert, we show that our agents are indistinguishable from human players.

CCS CONCEPTS

• **Computing methodologies → Artificial intelligence**; • **Applied computing → Computer games**.

KEYWORDS

Human-Agent Play, Chess, Game Playing Agents

ACM Reference Format:

. 2019. Playing Chess at a Human Desired Level and Style. In *HAI '19*. ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Mastering the game of chess is long believed to require a high level of intelligence [6]. It therefore comes as no surprise that the pioneers of Artificial Intelligence (AI) tried to develop computerized chess agents such as Shannon's [22] and Turing's 1951 'Turochamp' chess playing "agents" [8, Chapter 13] which were written on paper. Ever since, chess was intensely studied and several algorithms and heuristics were proposed including significant milestones such as Slate and Atkin's Chess 4.5 [26], IBM's Deep Blue [7] and, most recently, Deep Mind's AlphaZero [23].

In contrast to this impressive background of computational advances, which have already achieved super-human play level since the 90's [7], chess playing agents have also been adapted to play at an *adjustable level*, allowing for humans of different play levels to train with more "appropriate" automated opponents. These adjustments are commonly based on heuristics, targeted at limiting the agent's capabilities. Unfortunately, according to chess experts, existing agents still fall short of mimicking human play *style* (e.g., [16, 25, 27]), resulting in many human chess players preferring to train against human opponents instead of an automated agent in order to improve their game.

In order to address this shortcoming, we propose a novel approach for adjusting a chess agent's play level and style *simultaneously* by shaping its learned policy in a theoretically grounded manner. Our approach is built on the integration of the established theory of chess players' decision-making [9] with the state-of-the-art Monte Carlo Tree Search (MCTS) algorithm [5], commonly deployed by modern chess agents (e.g., AlphaZero [23]). Specifically, given a desired level of play, commonly provided by the Elo rating system [13], the agent's optimal play which was learned by the MCTS algorithm is systematically reshaped to resemble human play style.

In an extensive empirical evaluation, we demonstrate the benefits of our approach in three experimental settings: First, we train chess playing agents for various levels of play. The trained agents are evaluated via the popular human chess level assessment system Elometer[1] [11, 29] and their Elo estimations are consistent with the intended level of play. Second, we perform a tournament among the trained agents and show that their win-rate is consistent with that predicted by their estimated Elo rating. Last, in a Turing test-inspired human study, with the help of a chess expert and trainer (Elo rating of ~2100, who does not co-author this paper), we show that our chess playing agents are indistinguishable from human players whereas other leading agents are easily identified as non-human.

2 RELATED WORK

We discuss the two main research aspects of our task: the computational one and the behavioral one.

2.1 Computational Aspect

Developing game playing agents has been a prominent research theme in AI from its inception. This research has mainly focused on the task of achieving the "best" game playing capabilities, resulting in super-human performance in different game settings such as Chess [7], Checkers [19], Poker [4], Go [24] and others. Unfortunately, these agents often play in a distinctively different style than human players [14].

In order to design agents that are capable of playing at a human style, two methodologies are often deployed: 1) Imitation learning [28], where the agent trains to mimic human play by replicating available demonstrations or approximating them using supervised learning models; and 2) Domain-specific expert-based heuristics which are expected to result in a more human-like style (e.g., limiting look-ahead or simulation time, adding random noise to the agent's evaluation function, etc.). To the best of our knowledge, imitation learning was not applied thus far to the game of chess. This may be attributed to the enormous non-symmetrical state space of chess which limits the applicability of standard supervised learning approaches [1]. According to our chess expert, the use of

HAI '19, ,

© 2019 Association for Computing Machinery.

ACM ISBN 978-1-4503-9999-9/18/06…$15.00

https://doi.org/10.1145/1122445.1122456

[1] elometer.net

heuristics brings about low level of play and limited resemblance to human style. For example, an agent that was tuned for a specific Elo rating may indeed achieve the desired win-rate, however the individual moves vary drastically in the level: great moves coupled with occasional blunders.

Our proposed approach relies on modifying the popular MCTS algorithm in order to bring about a more human-like play. Similar efforts were recently made in various game settings such as the card game Spades [10], Candy Crush Saga [12], and general video games [15]. Common to these efforts is the heavy reliance on an imitation learning component that is trained using available human play data. As noted before, the use of imitation learning is unsuitable for the game of Chess. Other recent works in the Lords of War game [20] and Go [30] have also suggested techniques for biasing the MCTS action selection towards sub-optimal play. While the proposed techniques do not rely on any human generated data (and specifically do no use imitation learning), the techniques focus entirely on adjusting the agents' play with no explicit account for the agents' play style. To the best of our knowledge, this is the first work to explore the integration of human game-playing behavioral theories with MCTS and the first one to pursue human-like play, both in terms of level and style, in the game of Chess.

2.2 Behavioral Aspect

The cognitive abilities of Chess players have been studied for over a century starting from Alfred Binet's work [3]. In this work, we adopt the more modern, yet well-established, work of Adrian de'Groot [9].De'Groot claims that Chess players' thought process consists of two main components: *Intuition* (also known as orientation), and *Analysis* (also known as exploration, investigation and proof) [18].

An extensive evaluation of de'Groot's finding with human chess players reveals two phenomena: First, when chess players encounter a board position they almost immediately notice patterns rather than starting to expand many possible moves for all chess pieces (as classic chess agents do), constituting the "intuition" component of their decision-making. Specifically, more advanced players are better at recognizing "good" or "promising" patterns rather than simply performing a "deeper" thought process looking more moves ahead in the game.Based on their intuition, more advanced players are also more capable of performing "analysis" – namely, better exploring their intuition by "simulating" possible game traces in their mind.

In a similar fashion, the deep neural network used in Alpha Zero [23] provides an initial estimate of a win probability and a probabilistic preference over possible moves for any given board position (i.e., the intuition component). However, the network's 20 residual layers are able to capture much more than a human player can intuitively grasp. In addition, the clever use of MCTS guarantees an improvement over the use of the network alone by averaging future possible positions [2] (i.e., the analysis component), making the agent particularly successful against both people and automated agents. Due to its strong connections to Chess player's behavioral theories, in this work, we adapt the state-of-the-art Alpha Zero algorithm to better align with human-like style and level.

Since Alpha Zero's implementation and the trained networks are not in the public domain, we use the popular *Leela-chess-zero* (denoted LC0, lczero.org) implementation which mimics Alpha Zero's solution and has already achieved super-human chess capability. LC0 ranks among the very best chess programs today in international competitions.

3 APPROACH

The Alpha-zero agent consists of two main components:

(1) A trained neural network that provides for each board position an estimation of the win probability (*value head*), and a probability for selecting each of the possible moves (*policy head*).
(2) A modified MCTS algorithm, which incorporates exploration and stochasticity to improve the initial policy.

As noted before, Alpha zero achieves super-human level play. Degrading this level of play may be achieved by modifying one of the above components. We chose to focus on modifying the second component , i.e. the MCTS algorithm, and maintain the (near-)optimal trained neural network. Note that modifying the neural network for a specific level of play requires complex and time-consuming training, whereas our approach allows for flexible adjustment for multiple levels of play.

We leverage the three main hyper-parameters used in the Alpha-zero implementation:

(1) **Temperature,** τ: controls the selection of moves. When $\tau = 0$, the move with the highest visit count (N_i) is selected. Increasing τ increases the probability of selecting moves with lower visit counts, i.e. less promising moves. Formally, the move selection probability is given by:
$$\frac{N_i^{1/\tau}}{\sum_j N_j^{1/\tau}}$$
.
(2) **Policy Softmax temperature,** t: In the MCTS expansion nodes are selected based on the trained policy. Applying $t > 0$ promotes diversity by increasing the chance of selecting moves with lower policy probabilities (p_i). Formally, the expansion selection probability is given by:
$$\frac{p_i^{1/t}}{\sum_j p_j^{1/t}}$$
(3) **Exploration parameter,** c_{puct}: A constant that controls the trade-off between exploration and exploitation. Formally, for a state-action pair s, a (s-board position, a-move);
$$PUCT(s,a) = Q(s,a) + c_{puct} P(s,a) \frac{\sqrt{\sum_b N(s,b)}}{1 + N(s,a)}$$
Where Q is the mean action value (the average game result across current simulations that took action a) and P are the prior probabilities (as provided by the policy head of the network).

We denote the MCTS hyper-parameters as $\Theta = \langle \tau, t, c_{puct} \rangle$.

These role of the aforementioned hyper-parameters in the MCTS algorithm is analogous to the human behavioral aspects discussed in section 2.2. Specifically, τ is a technical interpretation of the "analysis" behavioral component: the higher τ is, the more "confused" the agent is with respect to the relative quality of each move derived via the MCTS. t is a technical interpretation of the "intuition" behavioral component: the higher t is - the more likely are less-promising branches of the MCTS tree to be explored. c_{puct}, the exploration parameter, can be seen as an enabler of both the intuition and analysis, since it emphasizes the exploration term which is controlled by the previous hyper-parameters.

We propose Algorithm 1 for tuning Θ given a desired level of play κ. Let D be a set of $\langle b, m, e\rangle$, where b is a board position, m is the associated move and e is the elo level of the player. In this work, we use the publicly available Lichess database [2].

First, we extract from D a subset $D_\kappa = \{\langle b, m, e\rangle : |e-\kappa| \leq \Delta e\}$ where $2\Delta e$ is the elo bin size. Next, we tune the hyper-parameters using numerical gradient descent [21] as described in Algorithm 1.

Algorithm 1 MCTS hyper-parameter tuning

```
1: repeat
2:     $\nabla\Theta \leftarrow 0$
3:     for each $d \in D_\kappa$ do
4:         $\overrightarrow{P(N)}, \vec{M} \leftarrow MCTS(\Theta, d.b)$
5:         $\vec{o} \leftarrow d.m$
6:         $\ell \leftarrow$ cross-entropy($\vec{o}, \overrightarrow{P(N)}$)
7:         $\nabla\Theta$ += numerical gradients($\ell$)
8:     $\Theta$ += $\alpha\nabla\Theta$
9: until convergence($\nabla\Theta$)
```

In words, in Lines 3 and 4 we run the MCTS algorithm given each sample $d \in D_\kappa$ board position as the root node. The MCTS output is a list of possible moves $\vec{M}$, ordered by their associated probabilities, $\overrightarrow{P(N)}$, which are derived from their respective visit counts ($\vec{N}$). Next, in Line 5, we translate the given move made by the player, $d.m$, into a one-hot vector, $\vec{o}$ with respect to $\vec{M}$. Namely, $\vec{o} = [0, \ldots, 0, 1, 0, \ldots, 0]$ where the 1 is at the index corresponding to the to the position of $d.m$ in $\vec{M}$. Then, the loss, ℓ, is calculated by the cross-entropy between $\vec{o}$ and $\overrightarrow{P(N)}$ in Line 6. In order to learn the MCTS hyper-parameters using stochastic gradient descent the gradients of the ℓ with respect to each hyper-parameter are needed. However, since the MCTS is non-differentiable, we resort to calculating numerical gradients in Line 7. This is accomplished by executing the MCTS algorithm *twice* for each hyper-parameter: $\forall x \in \{\tau, t, c_{puct}, \alpha, \epsilon\}$.

$$\frac{\partial \ell}{\partial \Theta.x} = \frac{\ell(\vec{(o)}, \overrightarrow{P(N)}|\Theta + \delta.x) - \ell(\vec{(o)}, \overrightarrow{P(N)}|\Theta)}{|\delta.x|}$$

Where $\delta.x$ is a small perturbation of the given hyper-parameter.

After the numerical gradients were accumulated in $\nabla\Theta$, in Line 8, we update the MCTS hyper-parameters following a standard gradient descent update rule with α as a learning rate. The process is repeated until convergence, namely $\nabla\Theta \approx 0$.

[2] database.lichess.org

4 EVALUATION

In order to evaluate our approach we first train four chess playing agents using Algorithm 1 for different levels of play. These four agents are evaluated using three experimental settings:

- **Experiment 1:** We perform a round-robin tournament between the agents to verify the agents' relative elo.
- **Experiment 2:** We test the agents' elo scores using a the popular chess level assessment system Elometer, especially designed for human players.
- **Experiment 3:** With the help of a chess expert and trainer, we again evaluate the elo levels of our agents. Furthermore, we perform a first-of-its-kind Turing test-inspired human study in which the expert has to distinguish between human and automated chess players.

4.1 Setup

Based on the United States Chess Federation (USCF) chess rating classes[3] we chose to focus on the following four classes:

(1) *Class D (elo 1200-1300):* a strong social player
(2) *Class B (elo 1600-1700):* above average tournament player
(3) *Expert (elo 2100-2200):* national expert or candidate master.
(4) *Grandmaster (elo 2500-2600)*: the highest international chess level class.

Using a month's games from the lichess database[4], we obtained 24,784,600 chess games, of which 2,772,575 games and 81,825,092 board position for Class D players, 4,481,130 games and 150,695,607 board position for Class B players, 1,026,468 games and 38,194,547 board position for Expert players and 44,585 games and 1,702,024 board position for Grandmaster players. These games were used to train four separate agents using Algorithm 1.

For the following experiments, the four agents were evaluated "as-is" without further tuning of their hyper-parameters.

4.2 Experiment 1:

In this experiment, we seek to verify the agents' elo ratings by performing a round-robin tournament between them. Note that the elo rating system further serves as a predictor of the outcome of a match between every two players. Specifically, following the common practice of chess tournaments, a win counts as +1 point, whereas a draw counts as +0.5 point for both players. It is therefor predicted that in a match between two players of with relative elo difference of $\Delta\text{Elo} = elo_1 - elo_2$ the expectation of the points earned by first player would be

$$\frac{1}{1 + 10^{\Delta\text{Elo}/400}}$$

In our setting, we expect the tournament to result in the win-rates provided in Table 1.

We used the CuteChess[5] to perform the experiment. Each pair of agents played 200 games against one another. The observed win-rate is reported in Table 2.

[3] https://bit.ly/2Vyl9Hi
[4] https://database.lichess.org/standard/lichess_db_standard_rated_2018-10.pgn.bz2
[5] https://cutechess.com/

	D	*B*	*Expert*	*GM*
D	-	0.091	0.006	0.001
B	0.909	-	0.053	0.006
Expert	0.994	0.946	-	0.091
GM	0.999	0.994	0.909	-

Table 1: The expected win-rate between the four agents given their trained elo.

	D	*B*	*Expert*	*GM*
D	-	0.105	0.002	0.000
B	0.895	-	0.039	0.005
Expert	0.998	0.961	-	0.065
GM	1.000	0.995	0.935	-

Table 2: The observed win-rate between the four agents given their trained elo.

Comparing the theoretical win-rate distribution with the empirical results using Pearson’s Chi-Square test [17] shows that the difference is attributed to chance with high probability ($p < 0.05$).

4.3 Experiment 2:

In Experiment 1, we show that the empirical win-rate in a tournament between the agents follows the theoretical one. However, recall that the win-rate depends **only** on the ΔElo, namely – the relative strength of each player. If we were to adjust the elo of all agents by a constant level, for example reduce all agent’s level by 200 elo points, the resulting theoretical win-rate table would be the same.

To address this shortcoming, we perform the following experiment: We evaluate each of our agents *independently* via Elometer system which is especially designed for human player’s level assessment. The Elometer presents 76 board positions to each agent, one after the other, requesting a move to be selected. From the selected moves, Elometer returns an estimate of the elo level. Table 3 summarizes the results.

	Trained elo	Est. elo
D	1200-1300	1316
B	1600-1700	1727
Expert	2100-2200	2108
GM	2500-2600	2546

Table 3: The EloMeter assessments compared to the trained elos of our agents.

4.4 Experiment 3:

Experiments 1 and 2 provide us with elo estimations for our agents. However, our task was not only to control the elo level of the agents but, and more importantly, bring about human play *style*.

To evaluate the playing style of our agents we perform a Turing-inspired human study. The experiment is designed to examine whether our agents can “fool” an expert trying to identify “non-human” play style. To that end, we use 3 types of players: 1) Human (extracted from the lichess dataset); 2) Typical agent (we used the leading Shredder program[6], which claims to be capable of adjusting its elo level in a human-like manner); and 3) Our agent, which we will denote as the Adjustable Chess Engine (ACE).

We created four sets of games each corresponding to one of the four elo levels discussed above. Each set consists of 18 games, divided equally into groups of 6 games, played between the same type of players: human-human, Shredder-Shredder and Ace-Ace. Each set of games was given to the chess expert who was asked to classify each game according to the style of play as either “human” or “computer”. Note that the expert was informed of the elo level of the set.

The results of our chess-expert’s classification are: 1) all of Shredder’s games were classified as “Machine”; 2) 1/2 of the human games were classified as “Human” whereas the other 1/2 were classified as “Machine”; 3) 1/2 of our agent’s games were classified as “Machine” and the other 1/2 were classified as “Human”.

This experiment validated our assumption that our agent’s play style is more similar to that of human chess players than the style of other chess agents, and in fact is indistinguishable from human players of varying levels.

5 CONCLUSION

In this paper we introduced a novel approach for training adaptable chess agents. We demonstrated that approach has the flexibility to match varying levels of human chess players. Moreover, our agent has the distinctive advantage of being able to play at a human style making it more suitable as a training partner for human chess players.

In future work we plan to perform more experiments, such as having a chess expert play games against unknown adversary (human or agent). Additionally, we plan to develop an interpolation function of our hyper-parameters, Θ, to enable a more fine grained level of play.

We also plan to deploy our agent in a local chess club, which will enable us to explore aspects such as modelling risk aversion of players, as well as varying playing styles in the different parts of the game: opening, mid-game and end-game.

REFERENCES

[1] Louis Victor Allis et al. 1994. *Searching for solutions in games and artificial intelligence.* Ponsen & Looijen Wageningen.

[2] David Auger, Adrien Couetoux, and Olivier Teytaud. 2013. Continuous upper confidence trees with polynomial exploration–consistency. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases.* Springer, 194–209.

[3] Alfred Binet. 1894. *Psychologie des grands calculateurs et joiers d’échecs.* Hachette.

[4] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. 2015. Heads-up limit holdâĂŽem poker is solved. *Science* 347, 6218 (2015), 145–149.

[5] Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. 2012. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in games* 4, 1 (2012), 1–43.

[6] Alexander P Burgoyne, Giovanni Sala, Fernand Gobet, Brooke N Macnamara, Guillermo Campitelli, and David Z Hambrick. 2016. The relationship between cognitive ability and chess skill: A comprehensive meta-analysis. *Intelligence* 59 (2016), 72–83.

[7] Murray Campbell, A Joseph Hoane, and Feng-hsiung Hsu. 2002. Deep blue. *Artificial intelligence* 134, 1-2 (2002), 57–83.

[8] B Jack Copeland. 2004. *The essential turing.* Clarendon Press.

[6]https://www.shredderchess.com/

[9] Adriaan D De Groot. 2014. *Thought and choice in chess*. Vol. 4. Walter de Gruyter GmbH & Co KG.

[10] Sam Devlin, Anastasija Anspoka, Nick Sephton, Peter I Cowling, and Jeff Rollason. 2016. Combining Gameplay Data With Monte Carlo Tree Search To Emulate Human Play. In *Twelfth Artificial Intelligence and Interactive Digital Entertainment Conference*.

[11] Birk Diedenhofen and Jochen Musch. 2018. An investigation into the usefulness of time-efficient item selection in computerized adaptive testing. *Psychological Test and Assessment Modeling* (2018), 289–308.

[12] Philipp Eisen. 2017. Simulating Human Game Play for Level Difficulty Estimation with Convolutional Neural Networks.

[13] Arpad E Elo. 1978. *The rating of chessplayers, past and present.* Arco Pub.

[14] Philip Hingston. 2012. *Believable Bots: Can Computers Play Like People?* Springer.

[15] Ahmed Khalifa, Aaron Isaksen, Julian Togelius, and Andy Nealen. 2016. Modifying MCTS for Human-Like General Video Game Playing. In *IJCAI*. 2514–2520.

[16] Monty Newborn. 2012. *Kasparov versus Deep Blue: Computer chess comes of age.* Springer Science & Business Media.

[17] CR Rao. 2002. Karl Pearson chi-square test the dawn of statistical inference. In *Goodness-of-fit tests and model validity*. Springer, 9–24.

[18] Philip E. Ross. 2006. The Expert Mind. *Scientific American* 295 (2006). Issue 2. https://doi.org/10.1038/scientificamerican0806-64

[19] Jonathan Schaeffer, Neil Burch, Yngvi Björnsson, Akihiro Kishimoto, Martin Müller, Robert Lake, Paul Lu, and Steve Sutphen. 2007. Checkers is solved. *science* 317, 5844 (2007), 1518–1522.

[20] Nick Sephton, Peter I Cowling, and Nicholas H Slaven. 2015. An experimental study of action selection mechanisms to create an entertaining opponent. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 122–129.

[21] Shai Shalev-Shwartz and Shai Ben-David. 2014. *Understanding machine learning: From theory to algorithms*. Cambridge university press.

[22] Claude E Shannon. 1950. Programming a computer for playing chess. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 41, 314 (1950), 256–275.

[23] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. 2017. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *arXiv preprint arXiv:1712.01815* (2017).

[24] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of Go without human knowledge. *Nature* 550, 7676 (2017), 354.

[25] Herbert A Simon and Jonathan Schaeffer. 1992. The game of chess. *Handbook of game theory with economic applications* 1 (1992), 1–17.

[26] David J Slate and Lawrence R Atkin. 1983. Chess 4.5âĂŤthe Northwestern University chess program. In *Chess skill in Man and Machine*. Springer, 82–118.

[27] Aaron Sloman. 1999. What sort of architecture is required for a human-like agent? In *Foundations of rational agency*. Springer, 35–52.

[28] Faraz Torabi, Garrett Warnell, and Peter Stone. 2018. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*. AAAI Press, 4950–4957.

[29] Han LJ Van Der Maas and Eric-Jan Wagenmakers. 2005. A psychometric analysis of chess expertise. *The American journal of psychology* (2005), 29–60.

[30] I-Chen Wu, Ti-Rong Wu, An-Jen Liu, Hung Guei, and Tinghan Wei. 2019. On Strength Adjustment for MCTS-Based Programs. In *AAAI*.