Hyperparameter Choice as Search Bias in AlphaZero

Eric M. Weiner
AMISTAD Lab
Harvey Mudd College
Claremont, CA, USA
eweiner@hmc.edu

George D. Montañez
AMISTAD Lab
Harvey Mudd College
Claremont, CA, USA
gmontanez@hmc.edu

Aaron Trujillo
Dept. of Computer Science
Harvey Mudd College
Claremont, CA, USA
atrujillo@hmc.edu

Abtin Molavi
Dept. of Computer Science
Harvey Mudd College
Claremont, CA, USA
amolavi@hmc.edu

***Abstract*—The AlphaZero algorithm has achieved remarkable success in a variety of sequential, perfect information games including Go, Shogi and chess. To better understand how AlphaZero works and leverage that understanding when deploying the system, we study the properties of the α hyperparameter that governs exploration noise in AlphaZero's search, the only hyperparameter the system's creators modified when moving among the three aforementioned games. First, we build a formal intuition for its behavior on a simple example meant to isolate the influence of the hyperparameter. Then, by comparing performance of AlphaZero agents with different α values on the game Connect 4, we show that the performance of AlphaZero improves considerably with a good choice of α. This all highlights the importance of α as an interpretable hyperparameter which allows for cross-game tuning that more opaque hyperparameters like model architecture may not.**

***Index Terms*—AlphaZero, Hyperparameters, Bias**

I. Introduction

AlphaZero is a general reinforcement learning algorithm designed to play games like Go and chess [1]–[3]. It has achieved superhuman performance on such games, without explicitly learning from any human gameplay [1], [4]. Yet AlphaZero still benefits from human insight for each of those games, because knowledge of the properties of good search coupled with an interpretable hyperparameter allows AlphaZero to be tuned in such a way as to play many different games well. Often hyperparameters come in the form of model architecture choices that can be opaque and challenging to optimize for a given problem. There is literature dedicated entirely to finding the correct settings of hyperparameters of AlphaZero [5], and stand-alone algorithms developed to tune the hyperparameters of AlphaZero and similar algorithms [6]. This kind of tuning allows for improved performance, but comes at a massive computational cost. Hyperparameters must be tuned from problem-to-problem, and this process quickly becomes prohibitively costly. As a result, researchers will often reuse hyperparameters hoping that the problems are "close enough" that changes are not necessary (e.g., [1]). The one hyperparameter in AlphaZero that *is* changed from game to game is the α value, since it can easily be adjusted to account for differences in game branching factors. This allows one to specify a trade-off between exploration and exploitation, which requires tuning for optimal performance. To gain better understanding of AlphaZero's workings, we investigate the influence of this hyperparameter in the system's search process, isolating its effects and building intuition for how to choose appropriate values without costly hyperparameter optimization.

This ability to tune based on intuition is powerful. In the earlier AlphaGo, hyperparameter tuning was a powerful tool in improving performance and achieving superhuman capability [7]. It is likely that many hyperparameters, including model architecture, are not optimal across the three different games of Go, Shogi, and chess; yet because of our inability to build intution around these hyperparameters, they are often left untouched. Prior work has explored empirical rules in special types of games, for example, a choice for different sizes of Go boards or a heuristic based on branching factors in games [1], [8]. In this paper we explore the intuition around the choice of the search parameter, examine the empirical differences in search with a Dirichlet prior, and finally perform a hyperparameter sweep that demonstrates that sufficient intuition allows for a strong choice of hyperparameter. We show that for searches with large enough branching factors, a Dirichlet prior that is appropriately aligned with information about the Monte Carlo Tree Search (MCTS) task could significantly improve performance over a uniform prior. We demonstrate the significant effect of α on search processes, increasing the probability of success on a simple synthetic problem that isolates its influence. This positive effect comes at the cost of needing correct bias to work well, as deeper search becomes more sensitive to the choice of α.

We are able to make an informed choice of α with intuition about what kind of search behavior is likely to benefit an agent on different types of games. This intuition acts as an implicit source of inductive bias in choosing α, and AlphaZero's success reflects a well-aligned inductive bias, as is the case for all successful learning and search processes [9]–[11]. While AlphaZero may not have had its parameters tuned as a result of human gameplay, the system still benefits from human experience in the form of well-aligned inductive biases.

II. Related Work

A. AlphaZero Overview

AlphaZero utilizes "self-play" and a variant on Monte Carlo Tree Search to play discrete, symmetric, two-player games [12]. One of the key pieces of the algorithm is the use of Monte Carlo Tree Search (MCTS), a search technique that has had success in Go going back to 2006 [13]. The key idea of the algorithm is to perform random exploration of a game while biasing exploration towards moves that are thought to have a good outcome. As a move is made more often in this exploration, the value of that node becomes more certain. At the end of the exploration phase, the action that was taken the most times during exploration is chosen as the move in the game. There are many ways to determine what is meant by a good outcome, and AlphaZero's MCTS utilizes a multi-headed neural network to predict values and future actions for different states in the game tree. In its search, AlphaZero stores and updates exploration probabilities at each node depending on a combination of several factors including the neural network policy output, results of simulated games, the number of visits to a given node, and an additional prior distribution affecting the root, controlled by the hyperparameter α. Like MCTS, AlphaZero's tree search is more likely to explore a less-visited or higher-reward state as determined by experience and the policy network [3]. The hyperparameter α influences the balance between the depth and width of exploration. Selection of α is an exploration vs. exploitation trade-off, and empirical evidence suggests that the optimal exploration-exploitation trade-off varies across different games [2]. This suggests that changing α can contribute significantly to success in AlphaZero.

B. Dirichlet Distributions

The parameter α is the parameter associated with a Dirichlet distribution. A Dirichlet distribution is a continuous probability distribution over the space of possible categorical distributions [14]. Generally, the Dirichlet distribution over β categorical variables is parameterized by a β-element vector of positive entries, $\boldsymbol{\alpha}$. In our case, the relevant family of Dirichlet distributions are centered about the uniform distribution, and thus every entry of $\boldsymbol{\alpha}$ is equal. Thus, we will use α to denote this scalar value in each entry. For this special case, the probability density function is given by

$$f(x_1, x_2, \ldots, x_\beta; \alpha) = \frac{\Gamma(\beta\alpha)}{\Gamma(\alpha)^\beta} \prod_{i=1}^{\beta} x_i^{\alpha-1}.$$

Intuitively, a large positive value of α indicates a high probability of selecting the uniform categorical distribution, $\alpha = 1$ indicates a uniform probability of selecting any categorical distribution, and a very small value of α indicates a high probability of selecting a categorical distribution that heavily favors a particular label.

C. Exploration in AlphaZero

Exploration in AlphaZero is partially driven by a Dirichlet distribution. AlphaZero's search is similar to that of Polynomial Upper Confidence Tree (PUCT) [15]. This strategy initially favors nodes with high prior probability and low visit count and asymptotically prefers high-scoring moves. In this algorithm, the action taken from each node is chosen by the formula: $a_t = \text{argmax}_a(Q(s_t, a) + U(s_t, a))$, with

$$U(s, a) = c_{puct} P(s, a) \frac{\sqrt{\Sigma_b N(s, b)}}{1 + N(s, a)}.$$

In the above formula c_{puct} is a hyperparameter to weight the trade-off between exploration and exploitation, $N(s, a)$ is a term that counts the number of times an action has been taken from a given state, and

$$P(s, a) = (1 - \epsilon) p_a + \epsilon \eta_a,$$

where $\epsilon = 0.25$ at the root, and 0 otherwise. We see ϵ is a hyperparameter that tunes the weight given to the search prior, and in AlphaZero the search prior was only used at the root node of the search for each move. The variable p_a denotes the output of AlphaZero's policy network and η_a is sampled from a symmetric Dirichlet distribution parameterized by α. In testing AlphaZero across different games, researchers changed the parameter α of the symmetric Dirichlet distribution, as shown in Table I along with each game's estimated branching factor β.

Game:	Chess	Shogi	Go
α	0.30	0.15	0.03
Approx. β	35	80	250

TABLE I
CHOSEN α VALUES COMPARED AGAINST THE β VALUES OF RELEVANT GAMES [3], [16]. THIS PARAMETER WAS THE ONLY MCTS PARAMETER VARIED BETWEEN GAMES IN THE ALPHAZERO PAPER.

In adjusting these hyperparameters for each specific game, the architects of AlphaZero were able to successfully inject a significant amount of inductive bias into their program based on an intuitive understanding of the parameter, and this inductive bias has been shown by Mitchell [17] and Montañez et al. [11] to be a necessary precondition for successful learning. We will further explore what this intuitively means in Section IV-A.

D. Hyperparameter Tuning in AlphaZero

There has been significant work in improving and updating AlphaZero from many different angles, including by tuning the parameters of the MCTS algorithm and the associated value and policy neural network as mentioned previously [5], [6]. Some of the original creators of AlphaZero and collaborators at Deepmind describe using Bayesian optimization techniques to tune the parameters of MCTS after training. This work is done on AlphaGo instead of AlphaZero, but the MCTS portion is quite similar between the two applications and it

seems reasonable to apply these ideas to AlphaZero [18]. Recently, there have also been various improvements to the AlphaZero algorithm, including modifying exploration incentives for moves with Dirichlet noise so that incorrect negative values for moves can more easily be overcome [19]. That said, there has been little deep analysis of the Dirichlet noise in AlphaZero, and the kinds of behaviors it may cause.

III. Favorable Bias Through Hyperparameter Tuning

A. Bias of Dirichlet

Research into search problems has shown that achieving better performance than uniform random sampling requires a bias that aligns with the underlying structure of a problem [10]. Without appropriate bias, search can do no better than uniform random sampling [11]. Viewing Monte Carlo Tree Search in this light, we can examine the bias in bits of a sampled Dirichlet prior as compared to a uniform prior. This bias caps the improvement a Dirichlet prior driven search can give over uniform search, and determines the proportion of problems this set of biases will do well on. The higher the bias, the more the algorithm is tailored to the specific search problem, and the less the general AlphaZero is with a fixed α. Bias, measured in bits, with respect to the hyperparameter α and the branching factor of the search tree β is given by

$$\text{Bias}(\alpha, \beta) = H(\mathcal{U}) - \mathbb{E}_{X \sim Dir(\alpha)}[H(X)]$$

where $\mathcal{U}$ is the uniform distribution, and $H(\cdot)$ is the differential entropy. The expected entropy of a categorical distribution drawn from a symmetric Dirichlet distribution is known to be $\mathbb{E}[H(X)] = \psi(\beta\alpha + 1) - \psi(\alpha + 1)$, where ψ is the digamma function [20]. Graphed over relevant branching factors and values of α, we see from Figure 1 that as α decreases the bias increases. Asymptotically, as α approaches zero, $\mathbb{E}_{X \sim Dir(\alpha)}[H(X)]$ approaches 0, so when α is sufficiently small, the bias is equal to the entropy of the uniform distribution for a fixed branching factor, or $\log \beta$. This means that a smaller α value represents a greater bias injected in the system, and so AlphaZero can only perform well on a small set of games with a fixed small α. This makes tuning α important to achieve good performance in a new application.

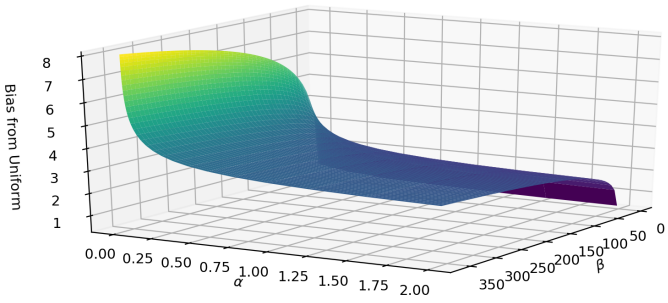

Fig. 1. Effect of β and α on Injected Bias. Notice for a fixed β, as α decreases, the bias from uniform increases. In this context, this means adding Dirchlet noise with low α represents significant information for how search should be conducted.

IV. Computational Results

A. Abstract Game Case Study

Armed with these insights concerning injected bias, we explore the how the choice of different priors can influence search and improve performance for different games and branching factors.

In order to try to isolate the influence of α on the search of a tree, we created a simple simulation that highlights the advantage of using a Dirichlet prior with specific α over a uniform prior in some types of games. We focused on trees that have sparse rewards several steps ahead in tree, which would be analogous to playing a game with sparse positive reward signals available only after several moves, which is representative of the types of games for which AlphaZero has had success. In this simulation, we assumed the game was a full tree of a certain depth, and a player could explore up to 200 nodes. At the specified depth, each node has a 5% chance of having reward 1, and otherwise has reward 0. At each non-leaf node, if the node is unexplored, a categorical distribution is sampled from a symmetric Dirichlet distribution with parameter α, and the next node choice is sampled from that categorical distribution. After exploring a node, the agent starts again from the root and chooses a path based on that node's categorical distribution, and continues recursively. If the algorithm visits a leaf with reward 1, it is considered successful. If the algorithm repeated a visit to a leaf, that was also considered an expansion in order to penalize revisiting parts of the game tree that do not represent reward. We tested 30 logartihmically spaced α values between 0 and 2 on different branching factors to find which choices of α lend themselves to successful searches for given branching factors and depth of reward.

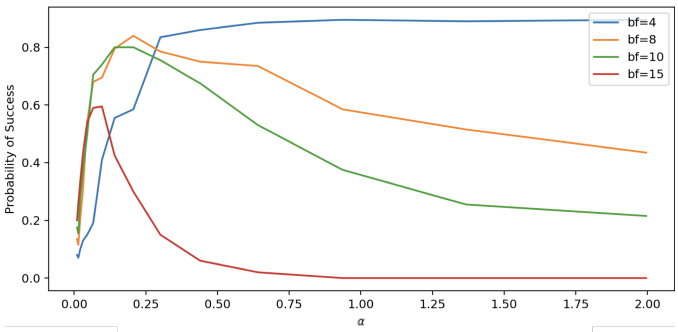

Fig. 2. Success Probability with Reward Depth 5 for different values of α. In this simulation, we observe that as the branching factor increases, there is a narrower band of α values that yield good results.

Figure 2 demonstrates the importance of aligning bias appropriately with the problem. These results and trends match intuition. For example, at small branching factors, we see that larger α values lead to a higher probability of success in this game. High α values correspond to a more uniform distribution sampled from the Dirichlet, and so this means that there isn't a need for large bias to succeed in this type of task for small branching factors. However, as the branching factor increases,

we see empirically that the range of α values that achieve good results decrease, and the best results come at smaller and smaller α. This indicates that at high branching factors the problem becomes harder and requires more bias away from a uniform distribution, making the selection of favorable α much more critical.

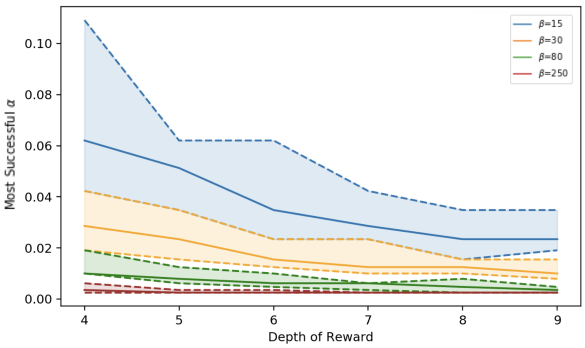

Fig. 3. Successful α Across Different Depths. The colored bands represent the inner 90% of 20 trials of exploration. This graph builds on intuition that as the depth of exploration required to get a reward increases, the most successful α decreases. In addition, as the branching factor increases there is a clear visual trend in decreasing the size of the band of successful α values.

Repeating this experiment across several different branching factors with rewards at several different depths, we found that in general as the depth of reward increased, the α value that led to the most successful search decreased (Figure 3), and as a result the mode depth explored for the most successful strategies increased. In Figure 3, the shaded regions around the solid-line averages capture the inner 90% of 20 trials, where each trial consists of 50 tree explorations across 30 α values between 0.0025 and 1, with the final score of a trial being the proportion of successful explorations to total explorations.

Fig. 4. Mode Depth Explored. As the depth of reward increases, we see a corresponding increase in the mode depth of exploration for the most successful α values. At high branching factors this depth of exploration is highly sensitive to the α value, and so we can build intuition as to why more precision in α is required for high branching factor games.

These results show that for similar games with high branching factor and sparser, deeper rewards, success is more sensitive to choice of the appropriate small α value, and a smaller α value corresponds to higher bias from uniform search.

Furthermore, as the average depth of the reward increased, the mode depth of exploration increased (Figure 4), and the corresponding successful α values were smaller (Figure 3) and more sensitive to changes. Since a lower α means greater bias (cf. Figure 1), higher branching factor problems and problems that require a deeper search require more biased search that is more sensitive to α values. While α values were more sensitive, this data suggests the intuition of the AlphaZero architects with respect to changing α to correspond with branching factor was a reasonable approximation. This seems to represent Silver et al.'s belief that occasional exploration to a depth of one or two from an action that may initially have the best prospect from the policy function is good policy [1]. In order to keep this property of AlphaZero across games, they were able to adjust accordingly.

B. Employing AlphaZero

We have shown that an appropriate choice of α significantly improves performance on a synthetic problem, and that it was relatively simple to predict values of α that are close to optimal. This was useful because it removed some of the complexity of AlphaZero and allowed for more controlled experiments, but does not fully resolve to what extent the choice of α improves search performance for AlphaZero. To show that these ideas transfer, we performed what amounts to a hyperparameter sweep on AlphaZero trained on the game Connect 4. If performance is improved by hyperparameter tuning on a simple game like Connect 4, then it is highly likely that α would be even more important in chess, Shogi and Go, as suggested by the results in Section IV-A. For training, we ran 200 descent steps for the main experiment, and 800 for the extended trials, where at each step the agent participated in 50 games of self play and loss was computed on a batch of 512 states from a buffer of 20,000 states. If this gradient step improved performance, the resulting agent was replaced. An implementation of AlphaZero with the configuration used to produce these results is available here: https://github.com/eweiner/AlphaZeroUncertainty.

C. Connect 4 Results

To evaluate the performance of different parameter settings, we sampled several different values of α, namely,

$$\alpha = \{0.01, 0.1, 0.2, 0.3, 0.5, 0.7, 1.0, 1.2, 10, 100\},$$

trained across ten network initializations per α, and played the agents against each other for four games each in a round-robin format. In these models, we explored 40 states in each MCTS, a significant change from the 800 states explored by AlphaZero in the paper, but not far off from the number of states explored in Atari games by MuZero [1], [21]. The results are included in Figure 5. These two plots represent the same results, but have two different scales for α. On the y-axis, we plot win rate. The alternatives are a draw or loss, but we feel win rate is most informative, as it represents true superiority over other models. As displayed in Figure 5, we found a win rate of about 45% as compared to an effectively

uniform baseline win rate of 32% ($\alpha = 100$). We also found that the biggest advantage was only in a small window of α values, which is consistent with the results of the abstract game. Noting that the difference is not as strong as was observed for the synthetic abstract game case, we attribute this to the fact that the noise was not the only exploration or exploitation signal in Connect 4, allowing agents to at least partially overcome a suboptimal initialization of parameters. These results support the conclusion that even though the root is the only node that directly gets the Dirichlet bias in Monte Carlo Tree Search, the performance of the algorithm changes significantly with different choices of α.

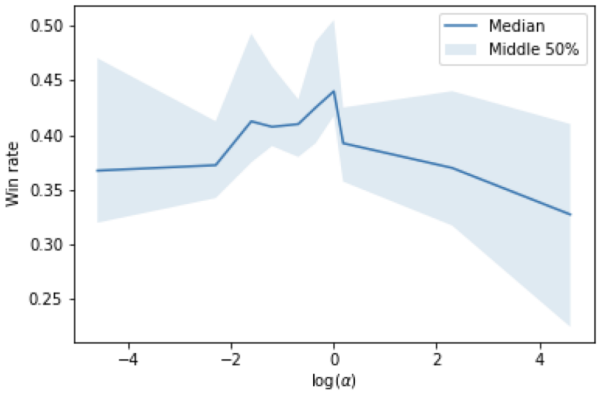

Fig. 5. Win rates across different values of α. The confidence interval represents the middle 50% of win rates. It should be noted that a significant number of games ended in a tie, and so a 45% win-rate does not mean that the agent lost 55% of games. In this graph we see a peak in performance around $\alpha = 0$. This supports the intuition that α would be an important parameter, but more tests are required for statistically significant results.

Beyond reinforcing ideas from the abstract game, the success of $\alpha = 1.0$ demonstrates that this parameter setting benefits from the intuition that a nonuniform exploration term helps AlphaZero find new states that will locally help the algorithm win a game, and at the same time increase training speed with proper exploration.

We halted training for each of these networks after 200 iterations, which was empirically observed to be a point of diminishing returns for the networks. That said, the loss value of the networks does not directly correspond to success against other agents, as exposure to new board configurations may temporarily cause the loss to increase. To ensure this did not significantly affect the results, two more tests were conducted. In the first, uninitialized networks were played against each other to ensure the effect of α did not start strong and then taper over time. The results of this test showed the highest performing α values were the smallest ones. This is likely because uninitialized policy and value functions perform like a uniform distribution in the synthetic abstract problem, where there would be in this case benefit to exploring the tree more deeply. In addition to the uninitialized network, ten networks were trained four times longer, for 800 iterations. Five of these networks had $\alpha = 0.7$ and five for $\alpha = 100$. These two settings repeated the format of playing all of the trials in the other setting for four games each. The $\alpha = 0.7$ agent beat the $\alpha = 100$ agent 45% of the time, while only losing 12% of the time and drawing the remaining 43% of the time. This suggests the trend identified by the larger study continues to hold, and in fact potentially gets stronger as training continues.

V. Discussion

Through these experiments we have shown that the the choice of Dirichlet prior can lead to a much greater rate of success, both in a reasonable, if simple, model of hard games like Go and chess, and in an actual AlphaZero system playing Connect 4. Even if one lacked an intuition for the proper setting of α beforehand, the synthetic abstract game provides a sense of ranges of α values that would be appropriate to elicit different search behaviors. Furthermore, since α appears in the exponent of the Dirichlet density function, we can surmise that searching on a logarithmic scale will be more informative than a more uniform search across α values. Experimentally, we see at this scale results are more symmetric, and operating at this scale we understand differences in the magnitude of α as more natural transitions.

The influence of α in AlphaZero is more complex than in our simple abstract game. For one, the noise is only applied at the root of each move, and the noise is only part of the exploration signal. We chose to add noise at each node of the abstract game because while noise isn't directly added to non-root nodes, exploration choices are made partially based on the policy network, which is trained from the final exploration distribution of the root node. While this effect isn't identical, the choice of α still has a global impact on the tree search.

Our results also show that for games with high branching factor, AlphaZero likely becomes more sensitive to the choice of α. This highlights the benefit that intuition about its behavior provides: for more complex games the number of good α values becomes quite small, and hyperparameter search becomes increasingly challenging.

AlphaZero is an impressive system, capable of learning superhuman strategy across several different games. We have shown ways in which the choice of a hyperparameter α can contribute to its strength as a system. Analyzing a synthetic abstract game, we demonstrate how to build intuition concerning the effect that exploration noise (parameterized by α) has on the search process, and suggest how one can use that intuition to avoid an uninformed hyperparameter sweep. Training AlphaZero systems is computationally costly; therefore, avoiding massively parallel tuning while still being able to tune the system from problem-to-problem can result in significant time and dollar savings. This understanding can lead to more custom-tailored deployments at reduced deployment cost. Understanding the inner workings of the system can also help us design new methods that leverage the same performance-enhancing properties.

A. Future Research

Future research directions are many. In this study, we investigate a way to find the expected effect of α choice on exploration within the context of a simple game. A full model could allow programmers to increase their intuition and select an expected distribution over different states at each level of search that they believe is targeted to the branching factor and problem attributes they are looking at. This work highlights the benefits of a more fundamental understanding of hyperparameters in general, because the current inability to easily tune systems from problem-to-problem may cause us to forego significant performance improvements. In this vein, a natural extension of this research is to build intuition and theoretical understanding of some of the other hyperparameter choices in AlphaZero, so that more general hyperparameter sweeps can be avoided in favor of targeted tuning.

REFERENCES

[1] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis, "Mastering the game of Go without human knowledge," *nature*, vol. 550, no. 7676, pp. 354–359, 2017.

[2] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2018. [Online]. Available: https://science.sciencemag.org/content/362/6419/1140

[3] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. P. Lillicrap, K. Simonyan, and D. Hassabis, "Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm," *CoRR*, vol. abs/1712.01815, 2017. [Online]. Available: http://arxiv.org/abs/1712.01815

[4] J. Ryan, "Scientists create AI that can crush the world's best AI (at board games, thankfully)," Dec 2018, `https://cnet.co/2Ge1wgW`.

[5] H. Wang, M. Emmerich, M. Preuss, and A. Plaat, "Hyper-Parameter Sweep on AlphaZero General," *CoRR*, vol. abs/1903.08129, 2019. [Online]. Available: http://arxiv.org/abs/1903.08129

[6] T.-R. Wu, T.-H. Wei, and I.-C. Wu, "Accelerating and Improving AlphaZero Using Population Based Training," 2020.

[7] Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas, "Bayesian optimization in alphago," 2018.

[8] D. J. Wu, "Accelerating self-play learning in go," 2020.

[9] C. Schaffer, "A Conservation Law for Generalization Performance," in *Machine Learning Proceedings 1994*. Elsevier, 1994, pp. 259–265.

[10] G. D. Montañez, "The Famine of Forte: Few Search Problems Greatly Favor Your Algorithm," in *Systems, Man, and Cybernetics (SMC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 477–482.

[11] G. D. Montañez, J. Hayase, J. Lauw, D. Macias, A. Trikha, and J. Vendemiatti, "The Futility of Bias-Free Learning and Search," in *32nd Australasian Joint Conference on Artificial Intelligence*. Springer, 2019, pp. 277–288.

[12] C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, S. Tavener, D. Perez, S. Samothrakis, S. Colton, and et al., "A survey of monte carlo tree search methods," *IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI*, 2012.

[13] R. Coulom, "Efficient selectivity and backup operators in monte-carlo tree search," in *Computers and Games*, H. J. van den Herik, P. Ciancarini, and H. H. L. M. J. Donkers, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 72–83.

[14] J. Lin, "On The Dirichlet Distribution," Master's thesis, Queens University, Kingston, Ontario, Canada, 2016.

[15] C. D. Rosin, "Multi-armed Bandits with Episode Context," *Annals of Mathematics and Artificial Intelligence*, vol. 61, no. 3, pp. 203–230, 2011.

[16] H. Matsubara, H. Iida, R. Grimbergen, and E. Laboratory, "Chess, Shogi, Go, natural developments in game research," *ICCA*, vol. 19, pp. 103–112, 12 1996.

[17] T. M. Mitchell, "The Need for Biases in Learning Generalizations," in *Rutgers University: CBM-TR-117*, 1980.

[18] Y. Chen, A. Huang, Z. Wang, I. Antonoglou, J. Schrittwieser, D. Silver, and N. de Freitas, "Bayesian optimization in alphago," *CoRR*, vol. abs/1812.06855, 2018. [Online]. Available: http://arxiv.org/abs/1812.06855

[19] D. J. Wu, "Accelerating self-play learning in go," 2020.

[20] I. Nemenman, F. Shafee, and W. Bialek, "Entropy and inference, revisited," 2001.

[21] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel *et al.*, "Mastering Atari, Go, Chess and Shogi by Planning with a Learned Model," *arXiv preprint arXiv:1911.08265*, 2019.