Paradox of AlphaZero: Strategic vs. Optimal Plays

Ze-Li Dou*
Department of Mathematics
Texas Christian University
Fort Worth, Texas 76109
z.dou@tcu.edu

Liran Ma*
Department of Computer Science
Texas Christian University
Fort Worth, Texas 76109
l.ma@tcu.edu

Khiem Nguyen†, Kien X. Nguyen†
Department of Computer Science
Texas Christian University
Fort Worth, Texas 76109
{khiem.nguyen, k.x.nguyen}@tcu.edu

***Abstract*—This article analyzes AlphaZero-type algorithms quantitatively from the viewpoint of local and global optimal sequences of play on a 7×7 board. Through targeted evaluation of the AI agent, the authors reveal the strategic, that is, winrate-dominated, nature of such algorithms, and expose thereby certain inherent obstacles against optimal play. Possible remedies are then explored, leading to techniques that may help further quantitative analysis of those algorithms and for the search for optimal solutions, on 7×7 as well as larger boards.**

***Index Terms*—Game of Go, artificial intelligence, AlphaZero, optimal play, targeted evaluation, targeted training.**

I. Introduction

AlphaGo [1] and its subsequent variations culminating in AlphaZero [2], notching victories against professional Go players for the very first time in history and yet achieving them with seeming ease, and eventually becoming self-taught from a near *tabula rasa* in the span of a mere couple of years, have attracted great attention not only in the field of artificial intelligence (AI), but also in a number of other fields spanning almost the entire spectrum of intellectual endeavors, from philosophy and cognitive science all the way to mathematics. Go players themselves seem to have completed a remarkable transition from initial skepticism to eventual reverence also: the Chinese professionals, for example, routinely refer to AlphaGo and AlphaZero as "Teacher Alpha", and their national team now features an official AI coach called *Jueyi* (the name has been translated to Fine Art in English) [3], which is built on the same principles of AlphaZero.

Has AlphaZero already reached perfection in the game of Go, or does it still have room for further growth? From the small number of published games, either self-played by AlphaGo or AlphaZero, or played between the two, we can infer that the answer is No. The differing margins of victory towards the end of the games tell us that conclusively.

The natural follow-up question, then, is how further improvement upon AlphaZero may be brought about. To answer this question, many researchers have chosen to emulate AlphaZero by adopting its basic approach but refining the technical aspects, so as to create a more efficient algorithm and, hopefully, a stronger playing agent. One could describe this general approach as trying to "speed AlphaZero up"; we shall have little to contribute in this regard except for a few comments towards the end of this paper.

In this article, we wish to discuss a rather different approach that is at once more conceptual and more quantitative. Currently, playing strength of the various AlphaZero-type algorithms is measured by their efficacy of scoring wins. Like the Elo rating system [4], this is a relative metric. However, there is a more absolute reference against which to define and measure strength as well, namely, *optimal sequences of play*. From that vantage point, the strength of an AI agent is measured by its ability to replicate or approximate optimal play. This is the approach we have taken in our experiments.

An immediate challenge in the abstract to this approach has to do with our lack of knowledge of what the optimal sequence of play is in any given game position. Since it is no longer a rare thing for an AI agent to play at a superhuman level, criticizing plays made by an AlphaZero-type algorithm could feel rather like a novice daring to gainsay his guru. For this reason, we have chosen to "slow down" AlphaZero rather than speeding it up. More precisely, we have conducted our investigations by applying an AlphaZero-type algorithm with commodity GPUs on a much smaller board (7×7). In so doing, the progress of the AI agent becomes more gradual, and positions may be set up where optimal plays can be demonstrated by humans. In addition, it is fortunate that global optimal solutions on the 7×7 board have been well studied by human players. As a result, we have been able to follow the (self-)training process of our AI playing agent, observe how far (and how frequently) it could go to follow global optimal solutions in its self-played games, and test its ability to discover optimal plays in various game positions specifically designed to target diverse stages of the game.

Winning plays and optimal plays are clearly related; they might appear roughly equivalent when viewed naively. Upon further consideration, however, the potential of an intriguing paradox emerges. As is well known, AlphaZero-type algorithms emphasize *winrate*, a probabilistic measure of how likely the game may be won. While it sounds vaguely logical that optimal plays should always be followed, this is not necessarily the case when one is behind; an aggressive play that leads to more complications may be seen as more strategic. By the same token, it is also possible to choose a suboptimal play strategically to avoid complications if one is in the lead.

Here we encounter a second, more practical, challenge. It

*The first two authors contributed equally.

†The last two authors contributed equally.

is well that we have a vague conjecture at hand, but to prove it rigorously with concrete scenarios is quite another matter. We have already mentioned the difficulty of knowing what the optimal line of play is, but even if it is known that an AI agent has not played optimally, how does one determine whether it is an instance of "strategic play" or simply one of failing to find the optimal play? To obtain clear-cut results, therefore, one needs game positions that hang on the knife-edge.

We have designed a large number of exercises for this very purpose, and this work is still on-going. A few representative examples will be exhibited and explained later in this article. Our experiments show that AlphaZero-type algorithms can be guilty of both overly aggressive and overly conservative plays for the sake of winrate. In fact, we shall show that the built-in strategic methodology may even *impede* the AI agent from playing optimally.

We hesitate to describe what we have uncovered as deficiencies of AlphaZero. Indeed, this line of AI agents have been widely praised for their "human-like" styles of play; perhaps strategic plays are partly responsible for such a feeling. Nevertheless, it goes without saying that understanding the difference between optimal and suboptimal plays is of great importance; it is so for the game's own sake, and from a scientific or philosophical standpoint, if we view Go as an exemplar of a deep problem with complex answers, this understanding is of fundamental significance. What we do claim, then, is that AlphaZero-type algorithms do not promote the discovery of optimal solutions as they are currently designed.

The rest of the paper is organized as follows. After a very brief discussion on existing and related work in Section II, we introduce basic concepts relevant to our work in Section III, and make precise the meaning of technical words such as optimal. We then demonstrate via concrete examples what we have qualitatively discussed thus far in Section IV; by probing the AI playing agent at well-targeted junctures in various stages of the game, we achieve a better understanding of the the workings behind the neural network of the AlphaZero-type algorithm. Not only so, we provide evidence that our techniques can help in our search for optimal plays as well in Section V. Some further general remarks, including a brief discussion of our project from a wider angle, are offered in Section VI.

II. Related Work

Stimulated by their dazzling successes, emulators of algorithms of the AlphaGo family are legion and comprise a very impressive international cast of researchers. Many of their creations have reached playing strength of strong professional players and beyond. In addition to Fine Art, already mentioned above, and Leela Zero, to be described below, there are also DeepZen Go of Japan, AI Handol of Korea, and ELF OpenGo of Facebook, to name just a very few. There is an equally brilliant array of innovations aimed at enhancing the efficiency and scope of AlphaZero-type algorithms. To touch upon just a small sampling of recent developments, KataGo [5], currently featuring arguably one of the best playing agents, has greatly accelerated self-play learning by introducing several improvements to the AlphaZero process and architecture, while the work presented in [6] has proposed a new model aimed at producing an agent that is strong (superhuman), stable and robust. See [7] and [8] for some other new advances.

We have instead decided to proceed in a different, one could almost say opposite, direction. Rather than striving for further enhancement in performance, we focus on the more foundational aspects of the issue and inquire into the learning and decision making processes of AlphaZero-type algorithms, to see whether or not there may be limitations in their capabilities, powerful as they certainly are. At the same time, we also concern ourselves with the quest of eventually discovering entirely optimal lines of play with the aid of the same algorithms. As we will explain, these investigations are not at all unrelated with one another. What needs to be emphasized here, however, is this, that if our work is in any way unique and relevant, it is only so in the context of all the spectacular developments brought about by other researchers.

III. Preliminaries

A. Go

The rules of Go are disarmingly simple. Two players, *Black* and *White*, take turns to place black and white stones on the intersections (points) of a square grid. The standard size is 19×19, but the game can be meaningfully played on any $n \times n$ board with $n \geq 3$. Except for their color, all stones are identical and have equal value; once played, they remain stationary unless captured. A group of one or more stones of the same color are *connected* if any two stones of the group can be linked by a chain of pairs either horizontally or vertically next to each other. A connected group of stones is *captured* (removed from the board as "prisoners") by the opponent, if all the intersections immediately next to the stones via horizontal and vertical lines (the *liberties* of the group) are occupied by stones of the opposing color; otherwise all the stones remain on the board. There can be no *suicide*—one is forbidden to place a stone at a point, if in so doing all the stones connect to that stone will have no liberty left as a group collectively. The exception to this rule is that such a play is legal if it is a capturing play. Finally, the overall board position can never be repeated; this is called the rule of *ko*. The basic rules for making legal plays are now complete.

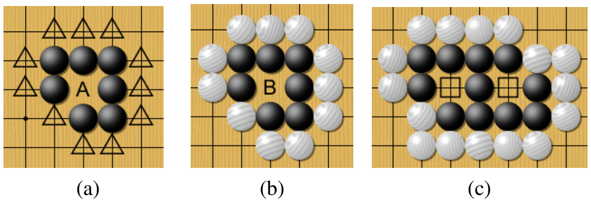


Fig. 1: Rules of Go.

These rules are illustrated by the figures below. In Fig. 1a, the group of seven black stones is connected with 12 liberties

(A and the points marked by triangles) in total. Black is allowed to play at A, since so doing will create a connected group with 11 liberties; but White is forbidden to play at A, since it would be suicide. On the other hand, by the same rules, it is permissible for White to play at B (it is a capturing play), but not so for Black (it would be suicide), as shown in Fig. 1b. The connected group of black stones in the Fig. 1c is permanently alive, because both points marked by squares are forbidden to White. When the entire board is occupied by living shapes, the player who has managed to control more than half of the board is declared the winner. In modern tournament play, a fiat called *komi* is added in order to compensate White, the player who plays second by convention. The scores of Black and White are compared after an agreed upon number, say 7.5, is added to that of White. The half point is designed to avoid ties.

The scores may be tallied as follows. The points on the board that belong to Black but are not occupied by black stones are Black's *territory*, which is to be compared with the territory of White, after the captured stones, along with the "dead stones" left behind in enemy territory, are returned to fill the opponent's territory—black prisoners filling Black's territory, and white prisoners filling White's. For an (overly) simple example, Fig. 2 shows a completed game on the 7×7 board. No capturing has occurred, nor are there any dead stones left behind. The intersections marked by triangles are Black's territory, 21 points; the squares mark White's territory, 14 points. Therefore, if the komi is at most 6.5 points, Black wins; if it is at least 7.5 points, White wins.

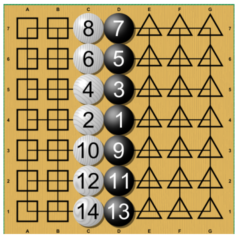


Fig. 2: A sample game.

For simplicity, we have omitted certain more nuanced scenarios of Go. In particular, we have not mentioned *seki*, which may be described as a symbiotic co-existence of "undead" stones. There also exist subtly different scoring systems, which may become relevant for us under certain circumstances, but not in this article.

It is well known that Go is extremely complex. The number of possible games in the first 50 plays exceeds the number of hydrogen atoms in the Universe. In more technical language, it is EXPTIME complete. Even the endgames phase is PSPACE hard [9]–[11].

B. Optimal Play

Given a particular size of Go board and a legal position on it, a legal play is said to be *locally optimal*, if it either maximizes the winning margin or minimizes the losing margin at the end of the game, assuming all subsequent plays are also locally optimal. Since the total number of possible games is finite, this definition is meaningful by reverse induction. A sequence of plays comprising an entire game and consisting of locally optimal plays only is called a *global optimal solution* for the given board size. Thus Go is strongly solved on the given board if all locally optimal plays are known, and weakly solved if all global optimal solutions are known. Words like local, global, and legal will oftentimes be omitted, if so-doing does not diminish clarity.

It is clear from the reversely inductive nature of the definitions that optimal plays and solutions are hard to find, and perhaps even harder to prove. For example, on a 19×19 board, it is almost certainly suboptimal to place the first stone at the very corner of the board. However, we are not aware of a mathematical proof of this.

On a square $n \times n$ board, all locally optimal plays are easy to find for $2 \leq n \leq 5$. For $n = 6$ and $n = 7$, the game has been extensively studied by professional Go players, such as Li Zhe [12]. For the purposes of this particular article, it is unnecessary to discuss the finer details regarding the status of the finality of optimal solutions; we may consider them known up to certain trivial exchanges of no material consequence. For $n > 7$ the game is currently unsolved.

C. The AlphaGo Family

The core of the AlphaGo family (e.g., AlphaGo [1], AlphaGo Zero [13] and AlphaZero [2]) is the combination of a convolution deep neural network f_θ and a Monte Carlo tree search (MCTS) algorithm. The parameters θ of the neural network represent Go knowledge, which are randomly initialized and iteratively trained by reinforcement learning from self-play (solely or combined with human knowledge). The neural network takes a game state s (board position) as input, and outputs a vector p_s (the *policy*) representing the *move probability* for each possible action and a numerical number $v_s \in [-1, 1]$ (the *value*, i.e., *winrate*) estimating the expected outcome z (-1 for a loss, 0 for a tie, and $+1$ for a win) based on s, as shown in Eq. 1.

$$(p_s, v_s) = f_\theta(s). \tag{1}$$

For both players, each play is selected by the MCTS algorithm, which contains a series of simulated self-play games (rollouts or playouts) that traverse a tree of states from root (the current state) to leaf. In each simulation, a move in each state is selected by evaluating the number of visits, the move probability, and the value (averaged over the leaf states of simulations) based on the current f_θ. The search reaches a leaf node after a number of selected moves down the tree. Then, it traces back (*backup*) along the traversed edges to update the number of visits and the value. After repeating the cycle of move selection and backup update for a number of times (e.g., $1,600$ in AlphaZero), the search returns a vector π representing the *search probability* distribution over possible moves, either proportionally or greedily with respect to the visit counts at the root state. The move with the highest π value will be selected as the next move.

At the end of each game, the terminal position is evaluated according to the rules of the game to compute the actual game outcome. The neural network parameters θ are updated so as to minimize the difference between the predicted outcome (winrate) and the actual game outcome, and to maximize the similarity of the policy vector to the search probabilities. Specifically, the parameters θ are adjusted by gradient descent on a loss function l that sums over mean-squared error and cross-entropy losses respectively, as shown in Eq. 2.

$$l = (z - v)^2 - \pi \log p + c\|\theta\|^2, \quad (2)$$

where c is a control parameter to regulate the level of weight regularization to prevent overfitting.

D. Experiment Configurations

Software tools. Our experiments are carried out using LeeLa Zero (v0.17) [14], which is an open source implementation of AlphaGo Zero. Since LeeLa Zero was designed for the board size of 19×19, we made a number of changes to make it suitable to play on smaller odd board sizes (e.g., 7×7, 9×9, and so on). Further, we utilize GoGUI (v1.5.1) [15] as the middle ware that enables LeelaZero engines to generate self-play games for training. We also use the neural network training script in TensorFlow (v1.13) provided in the LeelaZero repository to train our network. Lastly, we use Sabaki [16] to generate heat maps and analyze results.

Hyper-parameters. The major hyper-parameters used by the LeeLa Zero program for training our AI agents are listed in Table I.

Parameters	Definition	Value
$generations$	Number of generations	200
x	Number of random moves at the start of the game	10
c_{puct}	Exploration v. exploitation parameter	1.0
$playouts$	Number of playouts	150
$visits$	Number of visits	200
m	Number of games played per generation	6000
α	Number of past generations used for training	10
wr	Validation gate	55
b	Number of ResNet blocks	4
f	Number of ResNet filters	128

TABLE I: Major hyper-parameters used in our training.

Hardware. Our experiments are run on a machine with 32GB of RAM, 500GB of local storage, 1 Intel Core i9-9900K Processor (5GHz, 8 cores, 16 threads) and 1 NVIDIA TITAN RTX (24GB of memory). Each training run takes about 3 days.

IV. Targeted Evaluation

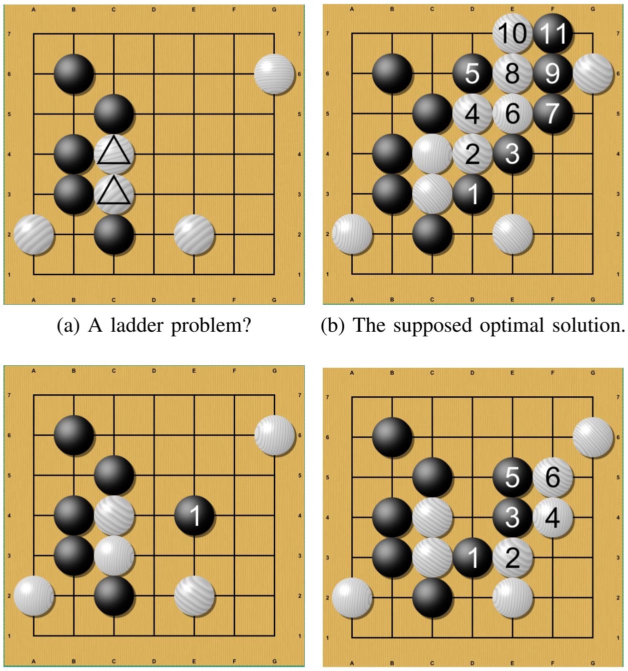
The self-played games by our trained AI agent do not follow global optimal solutions, even after the training is essentially "complete": that is, after its performance has plateaued. This is not surprising, since none of the Go-playing programs (e.g., KataGo [17]) we have examined online do so either. This fact alone is too vague to reveal all that much about the conceptual question we ask, namely, How does the AI "choose" its plays?

In order to extract more clear-cut answers, we tested our agent with numerous exercises specifically tailored for the purpose. Indeed, from the definition of optimality it is clear that a (not necessarily unique) locally optimal play always exists given any legal game position. This gives us a much wider range of freedom. We have dubbed this technique *targeted evaluation*, though in reality it is not an original idea at all—Go teachers have been training their human pupils for hundreds of years with basically the same technique.

Our findings will be illustrated below with a few representative exercises, selected to cover various phases of the game; experience with playing the game itself is not essential for comprehending the discussion. The examples will show that winrate exerts a dominantly strong influence over the AI agent's play selection from the very beginning of the game, so strong that the agent may select a suboptimal play in order to increase the probability of winning by just a little bit when behind, or similarly play suboptimally to avoid complications while ahead. Consequently, the AI agent is extremely sensitive to the amount of komi. This will be clear in most of the examples, but we shall emphasize this especially in the final example of this section to demonstrate that AlphaZero-type algorithms may impede the exhibition of global optimal solutions, even if so doing is within the capabilities of the AI.

A. Whole Board Thinking

A human pupil of Go usually learns the game one technique at a time: how to capture and avoid capture; how to achieve and prevent connection; how to make and destroy living shape, etc. A striking feature of an AlphaZero-type algorithm, however, is that concern for the advantage or disadvantage on the entire board is always the driving force behind its decision making. We had ourselves been surprised more than a few time by this.



(a) A ladder problem?

(b) The supposed optimal solution.

(c) The actual optimal play.

(d) Refutation of false solution.

Fig. 3: Whole board thinking.

The exercise shown in Fig. 3a is designed to test the AI's abilities at local fighting; more specifically, it tests the mastery of a basic capturing technique known as the *ladder*. Black is to play and capture the two white stones marked by triangles. The correct plays must frustrate all efforts by White to avoid capture. As such, the "optimal" sequence of plays is unique, as shown in Fig. 3b. After a quick review of Subsection III-A, the reader can ascertain without too much difficulty the fact that every play by Black threatens to capture the entire and growing white group with the next play—hence White's responses are all forced, but still fails to rescue the initial stone in the end.

To our great surprise, however, our AI agent failed to solve this exercise, which is considered relatively easy and a low-ranking human player is already expected to be able to solve it. The script we had written for detecting matching sequences of plays simply failed to find any. When we looked at the actual records, however, we discovered that the AI agent consistently made the play shown in Fig. 3c.

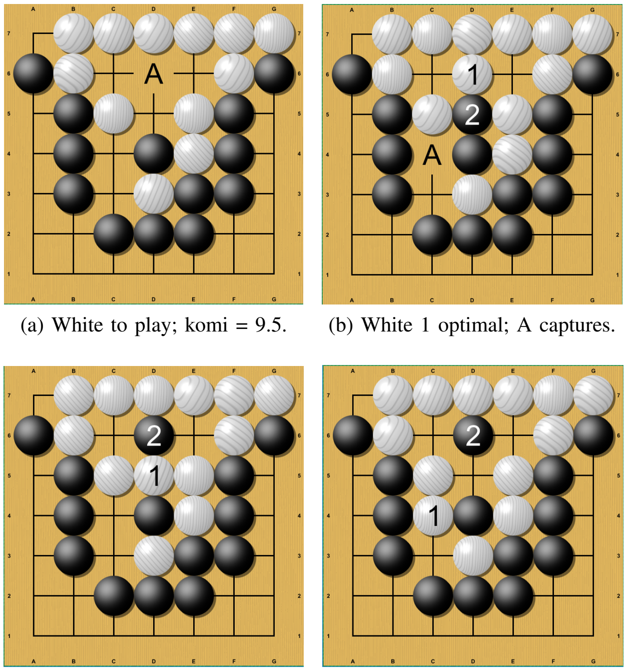
The truth is that the AI's play is superior to our so-called correct answer. Instead of narrowly aiming at the capture of a couple of white stones, the AI's play seeks to control the entire board. An experienced Go player will be able to verify easily that this play enables Black to claim the entire board—that is, all of the white stones will eventually "die", though none is immediately captured for now. By contrast, the "correct play" leaves White ample room to make permanent life for some of the stones. This is demonstrated in Fig. 3d. A reader inexperienced with Go may disregard Fig. 3d and take our assertion about Fig. 3c on faith.

In this example, our AI playing agent passes over immediate profit in a local fight in order to gain whole board dominance, since so doing gives it a more elevated winrate. Similarly motivated plays, however, can also cause the AI to err, as the next example shows.

B. A Winrate-induced Overplay

The optimal play for the exercise shown in Fig. 4a (White to move) is unique, for this is a so-called "life-and-death" problem, and White has just one group to save. More precisely, the white stones are surrounded, and there is only one play—at A—that can ensure permanent life for the group. This play simultaneously creates two points that are forbidden to Black; in Go terminology they are called *eyes*. Figure 4b shows how an attempt by Black at destroying White's eyes would fail, and Figs. 4c and 4d demonstrate that White's playing at any point other than A in Fig. 4a would not work. In both of the latter figures, the white group is "dead", that is, the stones can eventually be captured by Black if the game continues. In other words, Black would end up occupy the entire board.

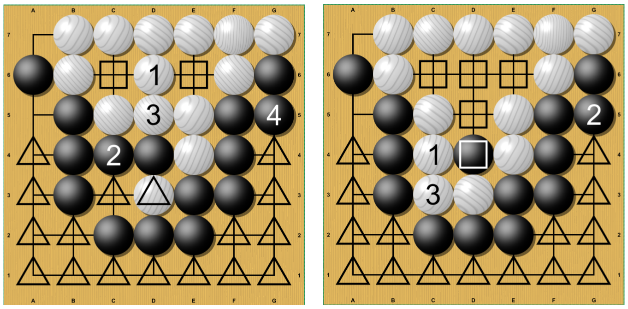
Once again, the AI agent surprised us during our targeted evaluation by stubbornly playing at 1 in Fig. 4d. We were rather puzzled until we learned to "think like AI" and adopted the whole board thinking as explained in the previous subsection.



(a) White to play; komi = 9.5. (b) White 1 optimal; A captures.

(c) Failure for White. (d) Another failure.

Fig. 4: A life-and-death problem.



(a) Optimal sequence rejected by AI. (b) Dream sequence for AI.

Fig. 5: AI's reasoning?

The komi (points assigned to White as a compensation for playing second; see Subsection III-A) was set at 9.5 in this experiment. Suppose White plays the correct—and this time it is also optimal—move, then the game will end as in Fig. 5a, with both sides playing optimally. We have also marked, by triangles and squares, territories occupied by Black and White. The result is then Black 17 territory + 1 dead stone = 18, White 2 territory + 9.5 komi = 11.5—a defeat for White. Now, if the wrong plays should miraculously work, as, say, Fig. 5b shows, White would end up with 5 territory + 1 dead stone + 9.5 komi = 15.5 points total versus Black's 15 territory points: White would win! Therefore, though 1 in Fig. 4d is an overplay, i.e., an overly aggressive play that can be refuted, it is a strategic one nevertheless. Perhaps that is why the AI considers it the right play. This is an amusing theory, but is there any chance of it being right?

We did not think so ourselves at first, not until we devised the exercise shown in Fig. 6, that is. Notice that the shape of the white group in the previous exercise appears exactly here as well, except that it is shifted to the right by a line. This is done to allow us to create a new group that is already permanently alive, at bottom left. The presence of this new group means that White has a winning game going, provided that the white group on the upper right does not die. And, lo and behold, though the technical problem has not changed at all, this time the same AI agent had no trouble whatsoever at finding the right, and in this case optimal, play!

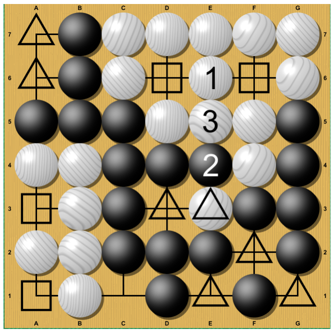
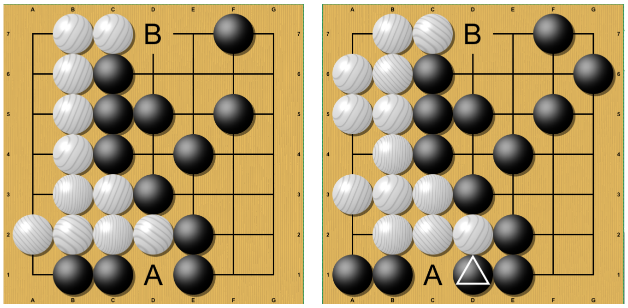


Fig. 6: AI plays optimally when in lead.

Summarizing, the same whole board thinking, driven by the same emphasis on winrate, has caused the AI agent to deliberately choose a suboptimal play, even though it is fully capable of finding the optimal play under the right—that is, winning—circumstances.

C. Avoiding Complication to Protect Winrate

It happens often that the optimality of a play is demonstrable only after the discovery of certain complicated lines of play. In the previous subsection, we have seen that the AI agent is willing to abandon a simple optimal play in exchange for an elevated winrate in an unfavorable position. We may surmise that the winrate-driven playing style of an AlphaZero-like playing agent may also strategically opt for a simpler line of play to avoid complications, even if it means abandoning optimality, when a winning game is at hand. This is indeed the case.



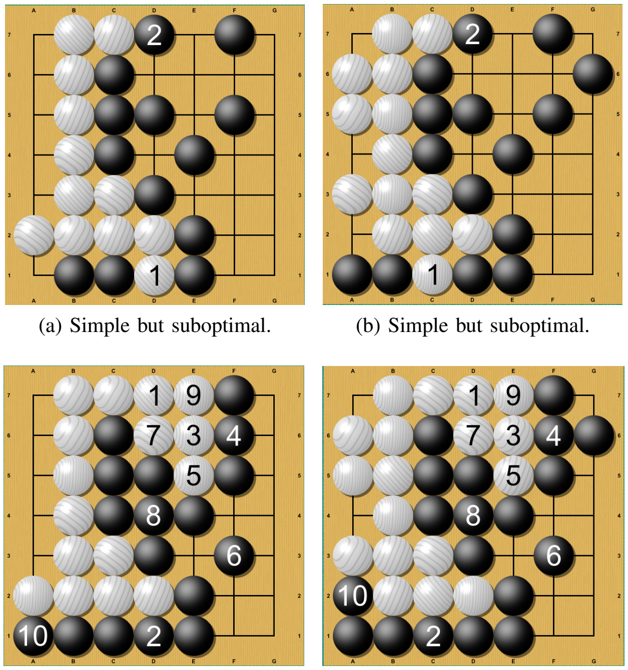
(a) White to play; komi = 8.5.

(b) A companion problem.

Fig. 7: A pair of endgame problems.

Unlike the “fighting” exercise in Subsection IV-A and the “life and death” exercise in Subsection IV-B, this time we offer an endgame exercise. Compare Fig. 7a with Fig. 7b. Black’s and White’s main groups are all safe in both exercises, and the focus is on finding plays that garner the maximal amount of points in terms of territory or, which is the opposite side of the same coin, to prevent the opponent from doing so. It is White’s turn to play in both exercises, the komi is set at 8.5 points, and the diagrams are rather similar. A close examination shows that the stone marked with a triangle in Fig. 7b gives Black a slight amount of established advantage over the situation in Fig. 7a.

Without giving quite as many details as we have done in the previous example, we point out that the choice is between playing at A, a simply play that secures immediate profit, and playing at B, a play that requires two follow-up plays to maximize its effect. In both cases, playing at B is the optimal play. Which line of play will our AI playing agent adopt?



(a) Simple but suboptimal.

(b) Simple but suboptimal.

(c) Optimal but complicated.

(d) Optimal but complicated.

Fig. 8: Optimality or simplicity?

By now the reader will not be surprised that the decision turns out to be determined by winrate. If we compare Figs. 8a and 8c, or similarly Figs. 8b and 8d, we see that the complicated line of play are superior. However, if we compare Figs. 8a and 8b, we discover a crucial difference: White wins by the minimal possible margin of 0.5 point in Fig. 8a, but loses in Fig. 8b. This decides White’s play: our agent chooses the simple play in Fig. 8a, but the complicated line in Fig. 8d. Rather uncanny!

To sum up, in this example we consider a pair of exercises that are extremely similar, technically speaking. In each exercise, a simple play suggests itself immediately, though a more complicated optimal solution also exists. The AI agent chooses the simple solution when it has a winning position, though it does adopt the more complicated line when behind and, furthermore, is quite capable of finding the optimal plays.

D. Strategic Play vs. Optimal Play

Our final example directly explores the relation between global optimal play and strategic play. If we pursue the chain of thoughts so far to its logical conclusion, the following conjecture is almost inevitable: given that an AlphaZero-type algorithm emphasizes winrate to a dominantly strong extent, it may in fact impede the exhibition of a global optimal line of play, since at some point in a self-played game, either Black or White would realize that the game is about to be decided with the given komi, and hence would tend to opt for a line of play either to protect the win or to fight for a chance at a reversal. In order to test the plausibility of such a conjecture as explicitly as possible, we conducted an experiment as follows.

First, we selected a specific game among the global optimal solutions in Li Zhe's article [12]. The margin of victory by Black, without komi, is 9 points. Next, we trained two playing agents from scratch under identical conditions, except that one has a built-in komi of 9.5 points, and the other 8.5. Finally, we gave as an exercise the game position that is 17 plays into of the optimal solution, which is shown in Fig. 9. In the same diagram we have also indicated the three plays that would lead to the conclusion of the optimal solution.

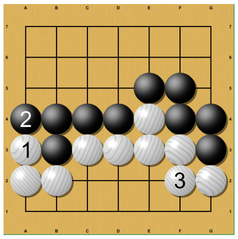
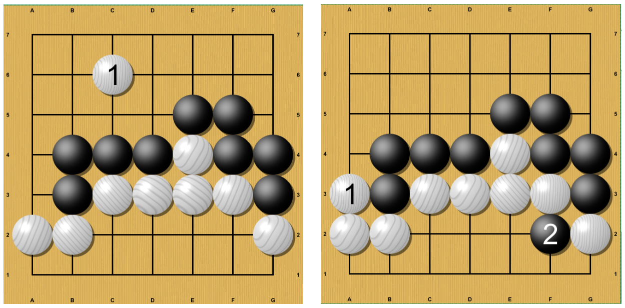


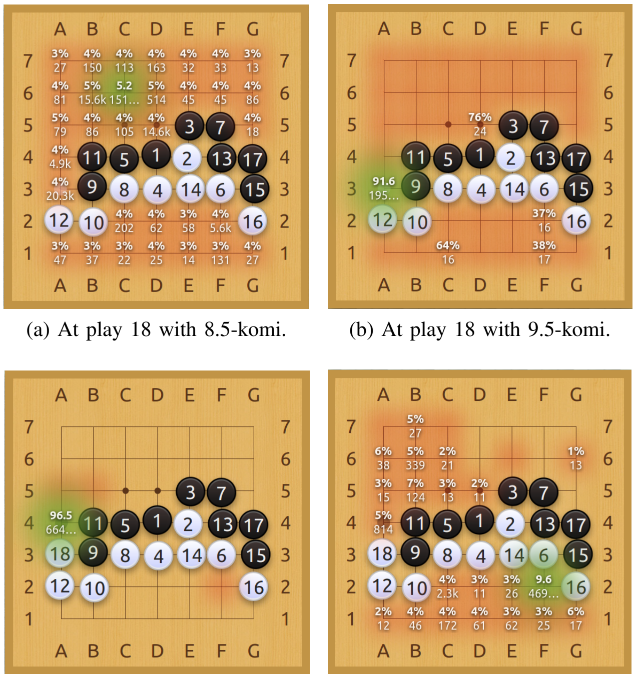
Fig. 9: Last three plays of an optimal solution.



(a) White overplays with 8.5-komi. (b) Black overplays with 9.5-komi.

Fig. 10: Suboptimal overplays.

Now the two playing agents either play themselves or against each other. Since the game is very near the end, White perceives that it is losing if komi is set at 8.5 points, and opts for a subnormal but aggressive play shown in Fig. 10a. When komi is 9.5, however, White is quite content to adopt the simple and optimal play in Fig. 9. Of course, we must remember that an AI playing agent is not human, and words and phrases like to "perceive", "to opt for", and "to be content" are but anthropomorphized descriptors of a vector p and a winrate v. In order to show the contrast with maximal clarity and also the aptness of our "feeling" in using such words, we have included the "heat maps" showing White's "decision making" as Figs. 11a and 11b. Likewise, in the 9.5-komi case, Black deviates from the optimal line with its own overplay in answer to White's optimal play; this is shown in Fig. 10b.



(a) At play 18 with 8.5-komi. (b) At play 18 with 9.5-komi.

(c) At play 19 with 8.5-komi. (d) At play 19 with 9.5-komi.

Fig. 11: Heat maps.

Consequently, *neither agent ends up producing the optimal solution in self-play*. However, when we let the 9.5-komi agent play White against the 8.5-komi's Black, the optimal line is generated with near certainty. The heat maps corresponding to Black's game analysis are given as Figs. 11c and 11d.

These results are explicit, consistent, and unequivocal. If its generalization to the 19 × 19 board holds true, the moral of the story is perhaps worth a rephrasing, since it is rather counterintuitive and hence a bit startling:

We may have never seen the best an AlphaZero-type algorithm can perform yet, and certainly not all of its best, because the winrate-dominated algorithm itself impedes optimal lines of play from being realized.

V. Targeted Training

An AlphaZero-type algorithm may be born with a near *tabula rasa* knowledge-wise, but we have seen in the previous section strong indications that it is nevertheless endowed with a "disposition" imposed by the winrate-related stipulations. This idea is natural, but it is much more difficult to quantitatively investigate, because it verges on the philosophical. Our current exploration is based on the hypothesis that, if the winrate-driven disposition of an AlphaZero-type algorithm should exert enough influence on the development of the playing agent to affect its trajectory, then one should be able to make the playing agent more "robust" by external steering—that is, dare we say it, by human intervention. We hasten to remark that, though it would be interesting for its own sake, our larger goal is by no means the making of another god-like Go playing agent. The ultimate aim should always be advancement towards a better architecture towards more generic artificial intelligence, and so it is for us. However, well designed human steering, if it should prove successful in enhancing robustness, is itself diagnostic of the areas where the current AI design may be improved upon.

As is well known, robustness is another large but vague word, but it at least allows the possibility of quantitative experiments. Our approach is to first identify areas our AI agent seems deficient, either in problem solving or in discovering possible optimal lines of playing, and then insert patterns tailored for these perceived weaknesses into the training of the AI agent. We have called this technique *targeted training*; a precise description of its meaning and implementation follows.

A. Implementation of Targeted Training

Targeted training is where puzzles or fixed openings are included in the self-play phase in order to guide the agent toward the "right direction". To realize this idea, in addition to playing against itself from the empty board, the agent also generated self-play games that start from an arbitrary number of openings as desired. The empty-board : opening ratio is defined as $e : o$, with $e + o = 1$. Regarding the network evaluation stage, there are two possible scenarios. If we target train on a set of openings and want to improve the overall strength of the AI, we pick $e > o$ and still compare the current best network and the newly trained one at the end of each generation. If we want the agent to focus on solving one (or few) specific puzzles, we pick $e < o$ for targeted training and skip the network comparison phase. We also train for a fewer number of generations since the initial network already has a solid amount of Go knowledge when trained traditionally.

Algorithm 1 describes the targeted training procedure in brevity (refer to Table I for parameters).

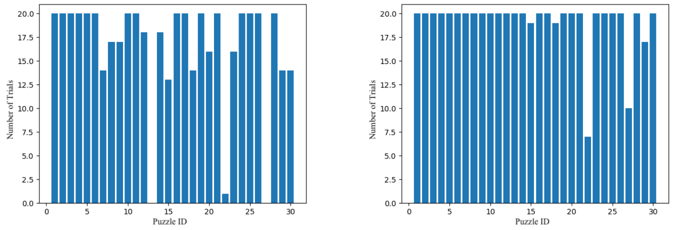
Algorithm 1 Targeted Training

```
$X^* \leftarrow$ initialize random neural network
$g \leftarrow 1$
**while** $g \leq generations$ **do**
    self-play $m * e$ games from empty board using $X^*$
    self-play $m * o$ games from openings using $X^*$
    train a new network $X$ on $\alpha * m$ games
    **if** $e > o$ **then**
        **if** $X$ is better than $X^*$ **then**
            $X^* \leftarrow X$
    **else if** $e < o$ **then**
        $X^* \leftarrow X$
    $g = g + 1$
```

B. Results

Our work in this area is still preliminary, though it appears hopeful. We mention two experiments briefly.

In one experiment, we assembled 30 exercises and incorporated them into our targeted training agent with the $e : o$ ratio set to $0.6 : 0.4$. Only the patterns were given in the training, not the solutions. Afterwards, we compared its performance in solving these exercises with another agent trained solely by self-play. Each agent was given 20 solving trials per exercise. The results are given in Fig. 12.



(a) Self-play-trained agent. (b) Target-trained agent.

Fig. 12: Problem solving by different agents.

The exercises are relatively easy, and both agents handled them rather well. However, the target-trained agent does perform somewhat better overall, especially in the case of a few exercises the self-trained agent finds nearly impossible.

In another experiment, we gave the first five plays in a global optimal game and incorporated it into the training of the AI agent. The AI had previously discovered the first five plays by itself already, and had followed them from time to time. The experiment simply enhanced its frequency in effect. Afterwards, we tested the target-trained agent and a self-trained agent with a position with six dictated plays, the five that were used for training, plus one play that neither agent ever followed. The result was interesting. Both agents discovered lines that are optimal almost to the end—but they are two different lines. Have we "altered the disposition" of the target-trained AI agent? Further experiments are needed before anything definitive can be said. The position we used for the experiment is shown in Fig. 13.

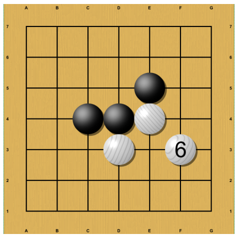


Fig. 13: The game position used for testing.

VI. Concluding Remarks and a Forward Glance

In this article, we have focused on the diagnostic aspects of our experiments. We have demonstrated that the winrate driven approach of an AlphaZero-type algorithm exerts a strong influence that permeates all phases of game play. Because of the fixed definition of what winning means (basic rules of the game plus a built-in komi), this influence may in fact discourage the global optimal solutions from being discovered—or at least exhibited—since the AI agent may deviate from optimal plays deliberately once it perceives the game position as being either favorable or unfavorable. On the other hand, we have also shown, convincingly we hope, that the AI agent is oftentimes quite capable of producing the optimal sequence if the conditions are right. In this sense an AlphaZero-type algorithm may produce an agent rather resembling an oracle: we often fail to get the right answers because it is hard to ask the right questions. This, then, is the paradox and enigma of AlphaZero. If this phenomenon has not been observed hitherto, it must be due to the fact that human playing strength has lagged behind that of the AI by too wide a margin.

We have anthropomorphized too much in an effort to avoid stilted language. One should always keep in mind that the AI algorithm does not think, or, at least, it does not "think" in the same manner we do. Nevertheless, it is perhaps not categorically incorrect for us to believe that, if the AI agent can play at superhuman level, the analysis of its neural network may involve sequences of play that are closer to optimal than humans have been able to produce by themselves. From this point of view, optimal plays and their approximations are of foundational importance for better understanding deep machine learning—they are more tangible than what one can garner, for example, from the AI algorithm's analysis of image recognition. AlphaZero is not explainable AI, but it leaves footprints, so to speak. Our interest in the search of optimal solutions either globally or locally, and, in fact, our interest in Go, ultimately lies therein.

In fact, the experiments and examples discussed here are part of an on-going project with a wider scope. Generally speaking, for example, we are interested in a more detailed analysis of the cognitive process of an AlphaZero-type algorithm. In working with a smaller-scale, "slowed down" algorithm, we hope to find a means to compare this process with the learning process of a "typical human", so to say. Targeted evaluation is systematically employed in this endeavor as we follow the progress of our playing agent through the generations of its training phase.

Another aspect of our project is the search for global optimal solutions. Here the issue is quite complex, since optimal sequences of play (local or global) may not be unique. Targeted training, which has been described here, though not in detail, is especially relevant in such cases. As we have explained, our oracle-like playing agent is not expected to voluntarily display all the possible lines considered by the network, but well-chosen dictated plays may bring out useful, perhaps even critical, information. Since the global optimal solutions can already be considered known on the 7×7 board, it is our hope that testing our methodology in that setting first will yield insight for a protocol for such an endeavor on larger boards. It is to be expected, of course, that the specific obstacles encountered by our particular AI agent may be easily overcome with the use of a more powerful computer. However, as we enlarge the size of the board, problems that the computer cannot immediately resolve will surely arise, regardless of the machine we use; a moment of reflection in terms of combinatorics will convince us of that. The real challenge, in fact, is to choose such targets judiciously so as to keep the amount of targeted training down to a manageable size. We hope to be able to report on these aspects of our experiment on another occasion.

Although we have chosen to present just one set of examples in this article in the interest of ensuring clarity and a sharp focus, we believe that our insight is already relevant, and our project shares with the other researchers on the AlphaZero-type algorithms the larger goal of better comprehending and utilizing deep machine learning through this remarkable advance in artificial intelligence. It is our hope that our approach will stimulate further research on the general subject, especially in hitherto less explored directions.

Acknowledgment

This work is supported in part by Texas Christian University (TCU) Invests in Scholarship (IS) grant. Liran Ma is supported in part by the National Science Foundation of the US under grant CNS-1912755.

References

[1] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, pp. 484–503, 2016.

[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.

[3] Tencent AI Lab, "Fine Art," 2017. [Online]. Available: https://tech.qq.com/a/20170319/015726.htm

[4] A. E. Elo, *The Rating of Chessplayers, Past and Present*. Arco Pub., 1978.

[5] D. J. Wu, "Accelerating self-play learning in go," *CoRR*, vol. abs/1902.10565, 2019. [Online]. Available: http://arxiv.org/abs/1902.10565

[6] F. Morandin, G. Amato, M. Fantozzi, R. Gini, C. Metta, and M. Parton, "Sai: a sensible artificial intelligence that plays with handicap and targets high scores in 9x9 go (extended version)," 2019.

[7] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, T. Lillicrap, and D. Silver, "Mastering atari, go, chess and shogi by planning with a learned model," 2019.

[8] B. Lee, A. Jackson, T. Madams, S. Troisi, and D. Jones, "Minigo: A case study in reproducing reinforcement learning research," 2019. [Online]. Available: https://openreview.net/pdf?id=H1eerhIpLV

[9] E. Berlekamp, J. Conway, and R. Guy, *Winning Ways for Your Mathematical Plays*. Academic Press, 1982.

[10] J. Tromp and G. Farnebäck, "Combinatorics of Go," in *Computers and Games*, H. J. van den Herik, P. Ciancarini, and H. H. L. M. J. Donkers, Eds. Springer Berlin Heidelberg, 2007, pp. 84–99.

[11] D. Wolfe, "Go endgames are pspace-hard," *More Games of No Chance*, vol. 42, 01 2002.

[12] Z. Li, "Qi Lu Qi Pan Zui You Jie (7x7 Go Optimal Solutions)," *The World of Weiqi*, vol. 20, Oct 2015.

[13] 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, pp. 354–359, Oct. 2017.

[14] G.-C. Pascutto and contributors, "Leela Zero," 2019. [Online]. Available: https://zero.sjeng.org

[15] M. Enzenberger, "GoGui," 2020. [Online]. Available: https://github.com/Remi-Coulom/gogui

[16] Y. Shen, "Sabaki: An elegant go/baduk/weiqi board and sgf editor for a more civilized age," 2020. [Online]. Available: https://github.com/SabakiHQ/Sabaki

[17] D. Wu, "KataGo," 2019. [Online]. Available: https://github.com/lightvector/KataGo