Can We Infer Move Sequences in Go from Stone Arrangements?*

Chu-Hsuan Hsueh[1][0000−0001−8888−3116]* and Kokolo Ikeda[1]

Japan Advanced Institute of Science and Technology, Japan
{hsuehch,kokolo}@jaist.ac.jp
*Corresponding author: hsuehch@jaist.ac.jp

Abstract. Inference commonly happens in our daily lives and is also a hot topic for AI research. In this paper, we infer move sequences in Go, i.e., the order in which moves are played, from stone arrangements on the board. We formulate the problem as likelihood maximization and employ a general optimization algorithm, simulated annealing, to solve it. Our experiments on professional and amateur games show that the proposed approach sometimes produces more natural move sequences than those played by humans.

Keywords: Inference· likelihood maximization· simulated annealing· Go.

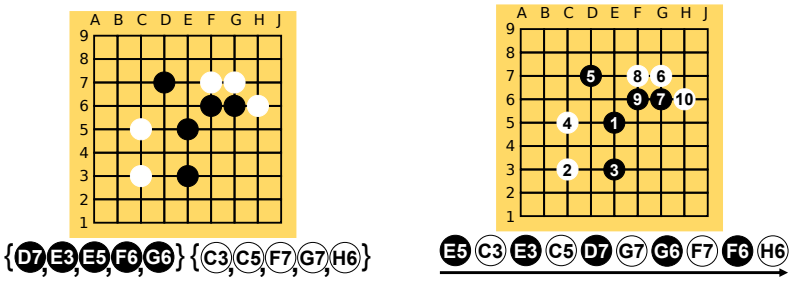
1 Introduction

Inference is a kind of intelligent work widely done by humans, meaning guesses or opinions based on known information. For example, detectives try to infer criminals and the approaches from pieces of evidence left at the scenes of the crimes. Doctors try to infer patients' abnormalities and the causes from indirect information such as symptoms and body checks. Inference also takes place in games. A typical example is to infer the cards on the opponents' hands in poker.

For the game of Go, strong human players can infer the move sequences at a glance of board states. There are two main reasons. First, most stones are on the board unless captured. Second and more importantly, strong players have played many games and know how games usually proceed. It is interesting to investigate whether and how computers can tackle such inference tasks.

In this paper, we formulate as a likelihood maximization problem the inference of a Go game's move sequence from stones on the board. An example on 9×9 boards is shown in Fig. 1, while we actually work on 19×19 boards. Assuming that the likelihood function of human players' moves is available, we treat the problem of finding the most likely move sequence as finding the move sequence with the highest likelihood. To simplify the problem, we target opening games without capturing stones in this paper.

* This work was supported by JSPS KAKENHI Grant Numbers JP23K17021 and JP23K11381.



(a) (b)

Fig. 1: Examples of (a) a stone arrangement and (b) a move sequence.

We employed simulated annealing (SA) [4], a general optimization algorithm often used for discrete search spaces, to solve our problem. We experimented with both professional and amateur players' games. We used a neural network trained using strong human players' games to approximate the likelihood of move sequences. The results showed that our approach inferred professional games better than amateur games and inferred shorter move sequences better than longer ones. Interestingly, for both professional and amateur games, our approach sometimes produced move sequences more natural than human players'.

The rest of this paper is structured as follows. Section 2 explains the problem formulation. Section 3 describes the background, including related work and SA. Section 4 presents our approaches for move sequence inference, and the results are shown in Section 5. Finally, Section 6 makes concluding remarks.

2 Problem Formulation

We formulate the move sequence inference in Go as a likelihood maximization problem[1]. The input is the stone arrangement on the board (e.g., Fig. 1a), and the output is the move sequence (e.g., Fig. 1b). First, we make three assumptions:

1. The likelihood function of human players' moves is available.
2. There are no handicap stones (i.e., black stones on the initial board).
3. No stones are captured.

The likelihood function in Assumption 1 is represented by a parameter θ, which outputs the probability that human players play a move m at a state s, denoted by $P_\theta(m|s)$, for all state-move pairs. For a move sequence (or history) $h = (m_1, m_2, ..., m_n)$ with n moves, the likelihood is calculated by $L_\theta(h) = \Pi_{i=1}^{n} P_\theta(m_i|s_{i-1})$, where s_0 is the initial state and s_i the state after playing m_1 to

[1] In some cases, players may intentionally play unexpected moves (e.g., to transpose to openings that are not well-studied so that the opponent may make mistakes). Such cases are not the inference targets in this paper.

m_i. Conceptually, the likelihood function can be of general human players, mid-level amateurs, or a specific player. Since it is hard to obtain the real likelihood in practice, we use an approximation for θ, which will be discussed in Section 5.

Assumptions 2 and 3 are made to simplify the problem. With Assumption 2, the initial state is an empty board. With Assumption 3, all stones are on the board after they are played; thus, we only need to consider coordinates with stones. In this way, the board state can be represented by two unordered sets of coordinates for black and white stones (Fig. 1a bottom) and a move sequence by an ordered list containing coordinates of black and white stones in turn (Fig. 1b bottom). Different move sequences have different permutations of coordinates, constraining that black and white stones are put one by another.

The problem of finding the most likely move sequence is then regarded as the problem of finding a permutation of stone coordinates with the highest likelihood. With n moves, the number of permutations is $(\lceil n/2 \rceil)!(\lfloor n/2 \rfloor)!$, indicating that this is a nontrivial problem.

3 Background

Subsection 3.1 discusses related work and Subsection 3.2 introduces SA.

3.1 Related Work

Researchers have applied likelihood maximization to solve various types of inference problems. For example, it has been widely used to reconstruct phylogenies in biology (the evolutionary history of organisms) [2]. Also, it has been applied to infer correct answers of tasks from the results of crowdsourcing platforms [1].

Some researchers built systems to record played moves in Go from videos (e.g., TV programs) [3,8]. They used image-processing techniques to identify boards and stones and then extracted moves when changes occurred. Their systems also generated move sequences from stone arrangements, but the problem and approaches differed from ours.

3.2 Simulated Annealing

Simulated annealing (SA) [4] is a stochastic optimization algorithm often used for problems of discrete search spaces. Starting with an initial solution, SA repeats the following processes for some iterations: (i) creating a new solution s' by a neighbor function that alters the current solution s a little, (ii) evaluating s and s' by a cost function that gives lower values to better solutions, and (iii) replacing s with s' with a probability of 100% when s' is better than s or a certain probability otherwise. The probability is $\exp(-\Delta E/T)$, where ΔE is calculated by the cost of s' minus that of s, and T is a temperature that gradually decreases over iterations. Users need to decide the initial and the final temperatures, the number of iterations, the solution representation (including initialization), and neighbor and cost functions, but nothing else.

4 Simulated Annealing for Inferring Go Move Sequences

The following explains our solution initialization and cost and neighbor functions. A trivial way to obtain an initial solution is to randomly decide the order of stone coordinates in the list. In addition to this, we propose a *greedy* method that uses the likelihood function as a heuristic. In more detail, we start from the initial state, select an unused stone of the player to move with the highest likelihood, play this move, and repeat this process until all stones are used.

Regarding the cost function, given a move sequence, we calculate its geometric mean of likelihood and define its cost as the inverse of the geometric mean so that better solutions have lower values. Calculating the geometric means is to make the cost function general to move sequences with different lengths. Otherwise, the likelihood usually gets smaller as the move sequence gets longer because the probability of playing a move at a state is usually lower than 1. Taking a move sequence with 20 moves and a likelihood of 10^{-40} as an example, the cost is $1/((10^{-40})^{1/20}) = 100$.

Neighbor functions in SA are usually designed to alter a solution a little, attempting to progressively improve the solutions through iterations. We propose 4 neighbor functions: **swp2**–randomly selecting 2 stones of the same color and swapping them (Fig. 2a), **mov2**–randomly selecting 2 consecutive stones and moving them to some other places while constraining that no stones of the same colors are put together (Fig. 2b), **rot3**–randomly selecting 3 stones of the same color and rotating them (Fig. 2c), and **mix**–using the above three alternately.

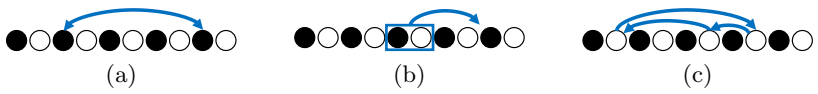


Fig. 2: Illustrations of neighbor functions: (a) swp2, (b) mov2, and (c) rot3.

5 Experiments

This section presents our experiments. Subsections 5.1 to 5.4 show the settings, the main results, selected move sequences, and ablation studies on SA options.

5.1 Experiment Settings

Regarding the likelihood function of human players' moves in Assumption 1, we employed a neural network (abbr. NN) trained using strong human players' games[2]. The NN takes the latest 8 states as inputs and then outputs the current

[2] https://sjeng.org/zero/best_v1.txt.zip from the Leela Zero project, https://github.com/leela-zero/leela-zero

state's policy (the probability distributions over moves) and value. We treated the policy output as the approximation of the likelihood function.

For game records used in the experiments, we chose professionals' and mid-level amateurs' games. Since the NN was trained using strong human players' games, we expected the network to be able to reflect the likelihood function of professional players' moves. For comparison, we intentionally included amateurs' games. It is interesting to investigate whether the NN is suitable for inferring weaker players' moves or whether the inferred move sequences look more reasonable than amateur players'. We selected 100 games for each set from openly available datasets[3]. The selected games did not contain handicap stones nor captured stones until the 50th move to satisfy Assumptions 2 and 3. We clipped the length to 20 and 40 moves from each game to test the scalability, while longer games remain as future work. To sum up, we had 4 sets of games: {professional, amateur} × {20 moves, 40 moves}, abbreviated by {Pro, Ama} × {20m, 40m}. Although we knew the actual move sequence in each game, the information was removed when making inferences. Namely, the inputs only contained the stone arrangements right after the 20th or 40th move was played (e.g., Fig. 1a bottom).

To infer the move sequence of a stone arrangement, we ran SA 5 times and took the move sequence with the highest likelihood as the solution. It is common to run SA several times because randomness is involved. We decided the parameters of the iteration number to be 5,000 and 20,000 for 20m and 40m games, respectively, and the initial and final temperatures to be 1.0 and 0.1, respectively, by some preliminary experiments. Initial solutions were obtained by the greedy method, and the mix neighbor function was applied.

5.2 Main Results

We first confirmed that SA worked well on the likelihood maximization for move sequence inference. We then investigated how much the inferred move sequences matched the actual ones. For some mismatched move sequences, we found that the inferred ones looked more reasonable than the actual ones. To analyze this, we quantified the goodness of moves. The following presents the details.

Likelihood Maximization To confirm whether the likelihood maximization was well done, ideally, we should compare the likelihoods of the inferred move sequences with the true maximum likelihoods. However, with 20 or 40 moves, each stone arrangement has $(10!)^2 \approx 10^{13}$ or $(20!)^2 \approx 10^{36}$ possible move sequences, making it infeasible to obtain the one with the maximum likelihood. Thus, we used the likelihood of the actual move sequences as baselines instead.

Fig. 3 shows the scatter plots of the geometric means of likelihoods, where the x-axis represents the actual move sequences and the y-axis the inferred ones. A point above $y = x$ means that the inferred move sequence has a higher likelihood than the actual one. Under Assumption 1, the inferred move sequence is more

[3] Professional: https://mega.co.nz/#!xFE2kTaK!Oj3_N9NpGmYVGTuka7Nc3T0HTmp3kKcXZR6p1Q7U5YU, amateur (1d): https://github.com/featurecat/go-dataset

likely played than the actual one. For the set of Pro20m, all points were on or above $y = x$. For the sets of Pro40m, Ama20m, and Ama40m, the ratios of points on or above $y = x$ were 65%, 93%, and 82%, respectively. We also obtained regression lines of the data points, constraining the intercepts to be 0. A slope higher than 1 indicates that the inferred move sequences generally have a higher likelihood than actual ones. The slopes of the 4 sets were 1.023, 0.988, 1.093, and 1.125. We concluded that the optimization was generally well done, though there was room to improve, especially in the Pro40m set.

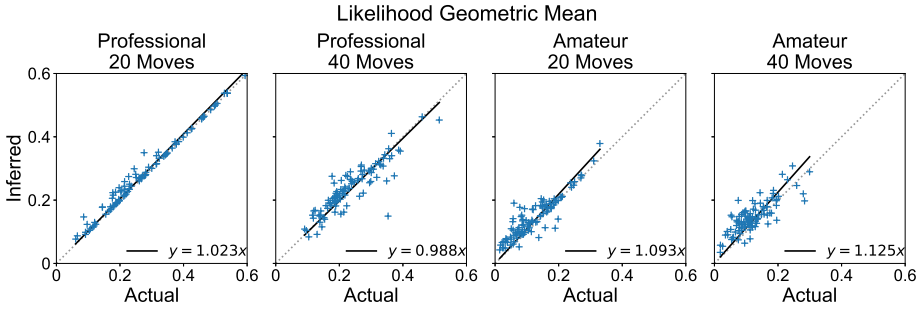


Fig. 3: Scatter plots of the geometric means of likelihoods: Pro20m, Pro40m, Ama20m, and Ama40m from the left.

Matching between Actual and Inferred Sequences After confirming that the likelihood maximization was generally well done, we investigated how much the inferred move sequences matched the actual ones. Among the inferred sequences, 51%, 2%, 11%, and 0% exactly matched the actual ones for Pro20m, Pro40m, Ama20m, and Ama40m, respectively. Longer sequences were more challenging to infer, as expected. In addition, it was also as expected that our approach matched Pro better than Ama.

With more moves, the matching rates decreased. Two possible reasons are discussed as follows. First, longer games, i.e., more complex solutions, made the optimization of SA more unstable. This was supported by the standard deviations of the likelihood geometric means of the 5 trials, which were 0.020, 0.028, 0.010, and 0.016 for the above 4 sets. Thus, SA might need more trials or tricks (e.g., advanced neighbor functions) to obtain a good solution. Second, when games proceed to the mid-game phase, it might be difficult for players to play good or natural moves. In contrast, in the opening phase, players can follow standard sequences (*joseki*), which is easier to match. Although it was not frequent for our approach to reproduce move sequences that exactly matched the actual ones, we found it successfully reproduced many sub-sequences.

The following discusses the differences between Pro and Ama, i.e., higher likelihoods (Fig. 3) and matching rates for Pro. We approximated the likelihood

function of human players' moves by an NN trained using strong human players' games. Thus, we considered it reasonable that the NN could better infer moves in Pro (who belong to strong players) than those in Ama. McIlroy-Young et al. [5] also reported that human players with different skill levels were best predicted by NNs trained using games of the corresponding level in chess. To make better inferences for the Ama cases, it is worth trying to train NNs for the corresponding skill levels. One potential problem we considered was that it might be intrinsically difficult to train an NN to well predict different amateur players of the same skill level because the players play a variety of moves (including bad moves). In this case, it is worth considering to model individual players [6].

Goodness of Moves Although the matching rates of the inferred sequences to the actual ones were not high in general, we found that some of the inferred sequences looked more *reasonable* than the actual ones when investigating mismatched move sequences where the likelihoods of the inferred move sequences were higher than the actual ones. Thus, we tried to quantify the goodness of move sequences by employing KataGo [10], one of the strongest open-sourced Go programs, for analysis. Territory advantage is one of the estimations output by KataGo, meaning the number of territory points that the current player is leading (e.g., +2.5 means that the current player leads the opponent by 2.5 points in territory). We defined the loss of a move m_i in a move sequence by $\delta_{maxi} - \delta_i$, where δ_i is the territory advantage of m_i and δ_{maxi} the maximum territory advantage among moves at the state before playing m_i. We then defined the loss of a move sequence by the average loss of the moves in the sequence. Move sequences with high losses usually contain some extremely bad moves, which may look unnatural. Thus, we considered move sequences with lower losses to be likely to look more reasonable.

Fig. 4 shows the scatter plots of the territory advantage losses, where the x-axis represents the actual move sequences and the y-axis the inferred ones. A point below $y = x$ means that the inferred move sequence has a lower loss than the actual one. Similar to Fig. 3, we obtained the regression lines of the points. In addition, we included the likelihood information in the plots. We calculated the ratio of the likelihood geometric means between the inferred and the actual sequences. A higher ratio means that the likelihood of the inferred sequence is much higher than the actual one. The ratios are represented by colors: higher ratios are closer to red, a ratio of 1 is white, and lower ratios are closer to blue.

The regression lines' slopes for Pro20m, Pro40m, Ama20m, and Ama40m were 1.002, 1.425, 0.903, and 0.929, respectively. A slope higher than 1 indicates that the inferred sequences tend to have higher losses than the actual ones (i.e., the inferred sequences with worse move goodness). Interestingly, the slopes for Pro were > 1 while Ama's were < 1. As shown in Fig. 4, some inferred move sequences in Pro had much higher losses than the actual sequences. In contrast, the inferred sequences in Ama had relatively close losses to the actual sequences, and some of the losses were much lower.

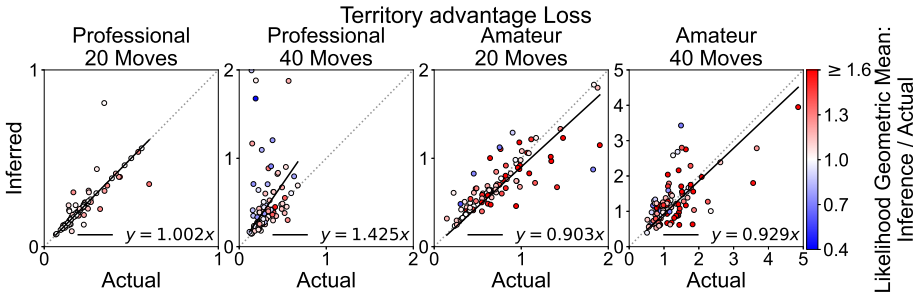


Fig. 4: Scatter plots of the territory advantage loss: Pro20m, Pro40m, Ama20m, and Ama40m from the left, where colors represent the ratio of the likelihood geometric means between the inferred and the actual move sequences.

Moreover, when looking at the colors together, we had two interesting observations, one from the Pro plots and the other from the Ama plots. In the Pro plots, some red or pink points were in the upper-left corners, especially in the 40m case. This means that the likelihood maximization was well done, but some extremely bad moves were produced in the move sequences. Since the goodness of moves was not the target of the maximization problem, it was not strange to have move sequences containing bad moves. As a future research direction, it is worth trying to reformulate the problem to also consider the goodness of moves (e.g., combining the policy outputs from NN trained using human players' games and NN trained using AlphaZero [7]).

In the Ama plots, many deep red points (the likelihood of the inferred sequence being much higher than the actual one) were below $y = x$ (the loss of the inferred sequence being lower than the actual one). We interpreted the results to be that our approach could produce more natural and reasonable move sequences than amateur players. Considering that amateur players originally played some bad moves and that the likelihood approximation was an NN trained using strong human players' games, we presumed it possible that good senses from strong players helped to avoid playing extremely bad moves.

5.3 Selected Move Sequences

For some move sequences where the inferred ones had higher likelihoods and lower territory advantage losses than the actual ones, including both professional and amateur games, we asked Go experts to point out which sequences looked more natural. The experts were not notified which sequences were inferred ones. Generally, the experts considered that the inferred sequences were more natural.

Fig. 5 shows 3 pairs where the experts evaluated the inferred sequences as more natural than the actual ones. The top pair was from Pro40m. The likelihood geometric means (abbr. LGMs) for inferred (left) and actual (right) were 0.262 and 0.206, respectively, and the losses were 0.301 and 0.557. The 1st to the 20th moves were the same. The experts commented that it was natural for Black

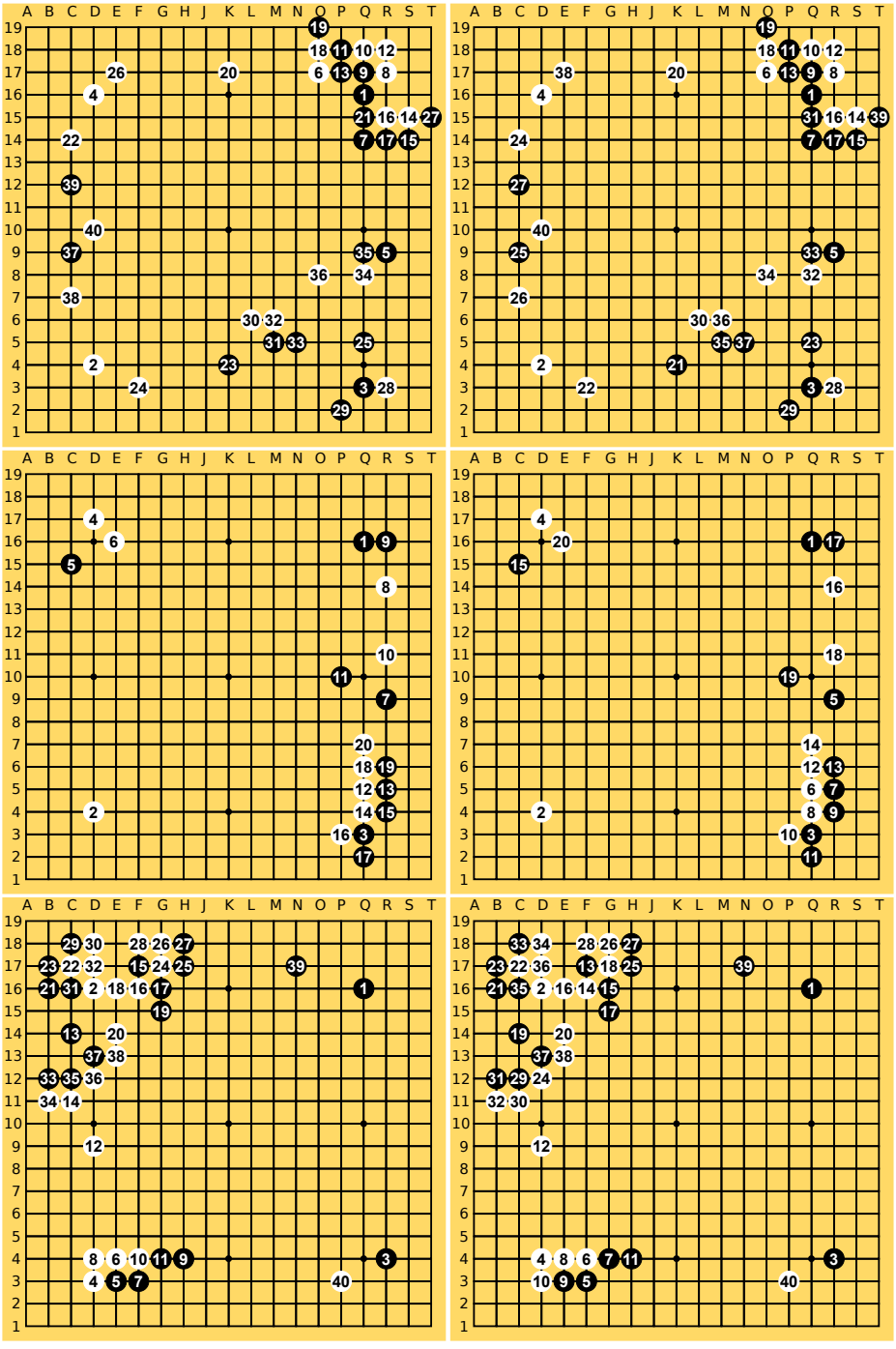


Fig. 5: Example move sequences: inferred on the left and actual on the right.

to play at Q15 to connect the black stones after White played at K17 as the 20th move. This made the shape of black stones thick. KataGo's analysis also showed that Q15 was urgent for both players. In the actual sequence, however, Q15 was omitted until the 31st move. In addition, the experts commented that the timing for Black to play at C9 and C12 to split white stones on the left was more natural after the enclosure of the upper-left corner by E17.

The middle pair was from Ama20m. The LGMs for inferred (left) and actual (right) were 0.153 and 0.084, respectively, and the losses were 0.460 and 0.941. The experts commented that the inferred sequence had more natural stone shapes and the direction of territory development. In contrast, for the actual sequence, the experts commented that the timings of the 15th, 16th, and 20th moves were strange and that the 19th and 20th moves were awful. For the 15th, 16th, and 20th moves, the probabilities from the NN's policy were 0.004, 0.002, and 0.001, respectively. For the 19th and the 20th moves, the losses were 3.673 and 3.150, respectively. Both statistics matched the expert evaluation.

The bottom pair was from Ama40m. The LGMs for inferred (left) and actual (right) were 0.199 and 0.075, respectively, and the losses were 0.814 and 1.292. For the inferred sequence, the experts commented that it was natural for Black to make *atari* (i.e., attempting to capture the opponent's stones) by playing at H17 as the 25th move, though it was a bit strange for Black to attach to the white stone at D3 by playing at E3 (the 5th move). The experts also commented that the flow was natural to make the black stones in the upper-left corner alive. In contrast, for the actual sequence, the experts commented that it was strange for White to make a pole connection by playing at E4 (the 8th move) and for Black to contact E4 from under by playing at E3 (the 9th move). The experts also commented that it was unusual for Black to ignore the cut at G18 from White (the 18th move).

5.4 Ablation Studies

We compared different solution initialization methods and neighbor functions proposed in Section 4. Since the results of Pro20m, Pro40m, Ama20m, and Ama40m were similar, we presented those of Pro20m for simplicity. First, regarding solution initialization, we compared the random and greedy methods. When evaluating the solutions by the highest likelihoods within 5 trials, the two methods did not differ too much. However, the stability differed a lot, where the standard deviation of 5 trials' likelihood geometric means was 0.020 for the greedy method but became 0.035 for random. We concluded that the greedy method helped stabilize the likelihood maximization. Second, regarding neighbor functions, we compared swp2, mov2, rot3, and mix. The slopes of the regression lines, like those in Fig. 3, for swp2, mov2, and rot3 were 0.828, 0.912, and 0.686, respectively (mix's was 1.023). When used individually, mov2 performed the best. Mixing different neighbors helped improve the likelihood maximization.

6 Conclusions

Inferring move sequences in Go from stone arrangements on the boards is a skill that many experienced Go players have. We formulated the problem as likelihood maximization and employed a neural network trained using strong human players' games as an approximation of the likelihood function. We adopted simulated annealing, a general optimization algorithm, to solve the maximization problem. We conducted experiments on opening games (20-move and 40-move) played by professional and amateur players. The results showed that the likelihood maximization was well done, though more complex problems (i.e., 40-move cases) still had room to improve. Only a few inferred move sequences exactly matched the actual ones; however, we considered that mismatches were not necessarily bad. We selected some mismatched sequences where the inferred sequences had higher likelihoods and lower territory advantage losses than the actual sequences, and we asked Go experts to evaluate these selected sequence pairs in a blind way. Generally, the experts judged that our approach produced more natural move sequences than those played by humans, including professionals and amateurs.

As interesting applications, our approach can be applied to amateur players as assistant tools. For example, it can be used to infer the move sequences of stone arrangements found in comic books or animations for fun or to infer those found in Go puzzle sets for further study. Regarding future research directions, first, it is interesting to compare move sequences inferred by humans with ours. Second, it is worth improving the scalability of the approach to longer games and more complex settings (e.g., move sequences containing captures). Third, it is promising to try Transformer [9], which is well known to be good at dealing with sequence data and has been applied to a wide variety of tasks.

References

1. Cui, L., Chen, J., He, W., Li, H., Guo, W., Su, Z.: Achieving approximate global optimization of truth inference for crowdsourcing microtasks. Data Sci. and Eng. **6**(3), 294–309 (2021). https://doi.org/10.1007/s41019-021-00164-2
2. Guindon, S., Gascuel, O.: A simple, fast, and accurate algorithm to estimate large phylogenies by maximum likelihood. Systematic Biol. **52**(5), 696–704 (2003). https://doi.org/10.1080/10635150390235520
3. Kang, D.C., Kim, H.J., Jung, K.H.: Automatic extraction of game record from TV Baduk program. In: The 7th Int. Conf. on Adv. Commun. Technol. (ICACT 2005). pp. 1185–1188 (2005). https://doi.org/10.1109/icact.2005.246173
4. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983). https://doi.org/10.1126/science.220.4598.671
5. McIlroy-Young, R., Sen, S., Kleinberg, J., Anderson, A.: Aligning superhuman AI with human behavior. In: Proc. the 26th ACM SIGKDD Int. Conf. Knowl. Discovery & Data Mining. pp. 1677–1687 (2020). https://doi.org/10.1145/3394486.3403219
6. McIlroy-Young, R., Wang, R., Sen, S., Kleinberg, J., Anderson, A.: Learning models of individual behavior in chess. In: Proc. the 28th ACM SIGKDD Conf. Knowl. Discovery and Data Mining. pp. 1253–1263 (2022). https://doi.org/10.1145/3534678.3539367

7. Ogawa, T., Hsueh, C.H., Ikeda, K.: Improving the human-likeness of game AI's moves by combining multiple prediction models. In: Proc. the 15th Int. Conf. on Agents and Artif. Intell. (ICAART 2023). pp. 931–939 (2023). https://doi.org/10.5220/0011804200003393
8. Shiba, K., Furuya, T., Nishi, S., Mori, K.: Automatic Go-record system using image processing. IEEJ Trans. on Electron., Inf. and Syst. **126**(8), 980–989 (2006). https://doi.org/10.1541/ieejeiss.126.980
9. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L.u., Polosukhin, I.: Attention is all you need. In: Advances in Neural Inf. Process. Syst. (NIPS 2017) (2017)
10. Wu, D.J.: Accelerating self-play learning in Go. In: The 34th AAAI Conf. Artif. Intell. (AAAI-20). Workshop on Reinforcement Learning in Games (2020), https://arxiv.org/abs/1902.10565