Analyzing the impact of knowledge and search in Monte Carlo Tree Search in Go

Farhad Haqiqat[1] and Martin Müller[1]

University of Alberta, Edmonton AB T6G 2R3, Canada
{haqiqath,mmueller}@ualberta.ca

Abstract. Domain-specific knowledge plays a significant role in the success of many Monte Carlo Tree Search (MCTS) programs. The details of how knowledge affects MCTS are still not well understood. In this paper, we focus on identifying the effects of different types of knowledge on the behaviour of the Monte Carlo Tree Search algorithm, using the game of Go as a case study. We measure the performance of each type of knowledge, and of deeper search by using two main metrics: The move prediction rate on games played by professional players, and the playing strength of an implementation in the open source program Fuego. We compare the result of these two evaluation methods in detail, in order to understand how effective they are in fully understanding a program's behaviour. A feature-based approach refines our analysis tools, and addresses some of the shortcomings of these two evaluation methods. This approach allows us to interpret different components of knowledge and deeper search in different phases of a game, and helps us to obtain a deeper understanding of the role of knowledge and its relation with search in the MCTS algorithm.

Keywords: MCTS · Go · Knowledge · Evaluation · Features

1 Introduction

Go programs achieved success first on small boards due to the power of Monte Carlo Tree Search, and later on the full 19×19 board due to the power of knowledge encoded in the deep neural networks. The MCTS based program Fuego [1] was the first program able to beat a top human professional player in Go on a 9×9 size board in 2008 [8]. Fuego achieved this level of play by using a MCTS algorithm enhanced by simple knowledge of features and patterns. Despite the successes that programs had on the 9×9 size board, the full 19×19 size board remained out of reach until recently, when AlphaGo [17,18] far exceeded the human level of play. AlphaGo uses a variant of MCTS with a very strong knowledge obtained through learning with Neural Networks (NN) in order to improve the search in MCTS.

Many studies have focused on increasing the strength of knowledge and methods of incorporating knowledge in MCTS based programs [5,4,6,16,20]. In order to examine the obtained knowledge and the resulting programs, move prediction

and playing strength are used as the most common evaluation methods. In this paper we investigate each of these methods and the impact of knowledge on the MCTS-based program Fuego.

In Section 2 first we briefly describe simple features. Then in Section 3 we introduce playing strength and move prediction as evaluation methods. We then describe the research motivations for this work in Section 4. Then in Section 5 we describe the experiments and analysis of the obtained result. In Section 6 we describe the conclusions and future work. This paper is a condensed version of the results in the first author's MSc thesis [12].

2 Knowledge and Simple Features

We define knowledge as information gained by training methods that helps a program to act in an informed manner, and improves the performance of a player when applied. One method of obtaining knowledge is by using features. Features are properties of a game state or a move, which can reveal information about that state or move. Examples of move features in Go are whether it captures any stone, or creates a ko situation. Coulom [6] defined a set of simple move features that were extensively used and extended by other programs. In order to use simple features, we need to evaluate them. Fuego uses Latent Factor Ranking (LFR) [16] as its evaluation and training method for simple features.

3 Evaluation Methods

3.1 Playing Strength

One method for testing knowledge is comparing playing strength with and without the use of that knowledge. In this scenario we use knowledge either as a standalone player or integrate it into an available program, and play a number of matches against other programs or another version of itself. If we know the level of strength of the opponent, then we can estimate the strength of our program from the obtained results by using the Elo rating formula [7]. If we integrate knowledge in an existing program, then we can estimate the quality of the knowledge by measuring the increase in playing strength resulting from the added knowledge.

3.2 Move Prediction

Move prediction is the act of predicting the next move in a game that was played before. To predict a move, we select a position from a game and feed that position to the game-playing engine. Then we compare the response received with the next move played in the game. Games played by professional players are one of the main data sources for training knowledge in Go. In order to evaluate the obtained knowledge move prediction is very often used.

Coulom [6] reported a 34.9% prediction rate for his feature-based knowledge learning technique. Wistuba et al. [16] were able to reach 40.9% prediction rate. Xiao et al [20] were able to improve Wistuba's [16] method to increase the prediction rate by 2%. Clark et al. [5] used deep neural networks to increase the prediction rate to 44%. Soon after, Maddison et al reached 55% with a deeper and larger network [15]. The current state of art in move predictors are deep residual networks [13]. Cazenave reports a 58% prediction rate with such an approach [4]. In general, stronger knowledge has led to stronger play. We want to measure whether in different Fuego-based players, increases in prediction rate correspond with increased playing strength.

4 Research Motivations

The relation between knowledge and search in Go programs and how these two impact each other is an area that still needs more study. Research motivations for this work are as following: 1. *Examine current evaluation approaches used in Go programs, which are: move prediction and playing against another program. Understand the differences between each of these tests and how they relate to each other*, 2. *Evaluate the impact of knowledge on the performance of a Go program*, 3. *How does longer and deeper search improve the strength of a MCTS program, in the presence of knowledge?*, and 4.*Can this increased strength be explained in terms of simple feature knowledge?*

Many researchers have studied simulation policies in Go, focusing on move prediction as an evaluation method, or comparing the strength of two programs [10,14,19]. Xiao et al. [21] report both improved move prediction and playing strength of Fuego after adding stronger knowledge; however, no insight is given to what those changes in the evaluation mean. Fernando et al. [9] analyze the Fuego simulation policy and the impact of changes to it. No analysis was done on evaluation methods and interpretation of the effect of added knowledge. Our current work is the first such analysis.

5 Experimental Results and Discussion

In this section we first we describe the players used in our experiments, then provide the results. We explain those results, and use them to investigate our research motivations.

5.1 Fuego-Based Players Used in our Experiments

in our experiments we have used the following players from the Fuego code base.

Playout Policy-Only Player: this simple player uses only the playout policy of Fuego for generating the next move in the game, and it does not use search. This player helps us to understand the playout policy in Fuego better, and also helps us to measure different aspects of the playout policy, such as move prediction and playing strength.

Simple Features-Only Player: here, we use the prior knowledge in Fuego as a stand-alone player. The highest evaluated move according to features is played. Having a features-only player helps us to understand how the knowledge encoded in features compares to search, and it also helps to better evaluate feature knowledge.

No Knowledge Player: in order to examine how knowledge helps the performance of a player, we turn off prior knowledge and move filtering in Fuego. This player uses only MCTS with the default Fuego playout policy. This player helps us better understand the impact of search on a player's move prediction and playing strength.

No Additive Player: Fuego by default uses additive knowledge to help its in-tree policy focus more on high-ranking moves. We turn off the additive knowledge in this player, and rollback Fuego to use MCTS with the UCT method. This player helps us to better understand the role of additive knowledge in Fuego.

Default MCTS-Based Fuego Player: we need to be able to compare the results obtained by other players with full-strength Fuego. This player uses the full Fuego engine with all default settings.

Varying the Number of Simulations: for the MCTS-based players No Knowledge, No Additive and Default we vary the number of simulations in {100, 300, 1000, 3000, 10000}. This helps us to understand the impact of more search on different players.

5.2 Move Prediction

For the move prediction task, we used games from Pro Game Collection [3]. In total we used 4621 games, after removing games that were played on board sizes other than 19×19. Table 1 shows the results of the move prediction task on all the positions from these games. The players are Fuego-based engines described in Section 5.1. The move prediction rate is the fraction of positions for which the professional move was predicted correctly. For the No Knowledge, No Additive, and Default Fuego players the number in the name represents the number of simulations per move used by that player. Figure 1 shows the prediction rate for different number of simulations.

The Playout Policy-Only and Simple Features-Only players do not use Monte Carlo simulations. Playout Policy-Only predicted less than 22% of professional moves. Simple Features-Only has a much higher prediction rate of approximately 31%. Given the fact that neither of those two players uses MCTS, the gap signifies the role of the knowledge obtained through a large set of simple features trained by machine learning methods in the Simple Features-Only player, compared to the combination of fast rules and small patterns in the Playout Policy-Only player.

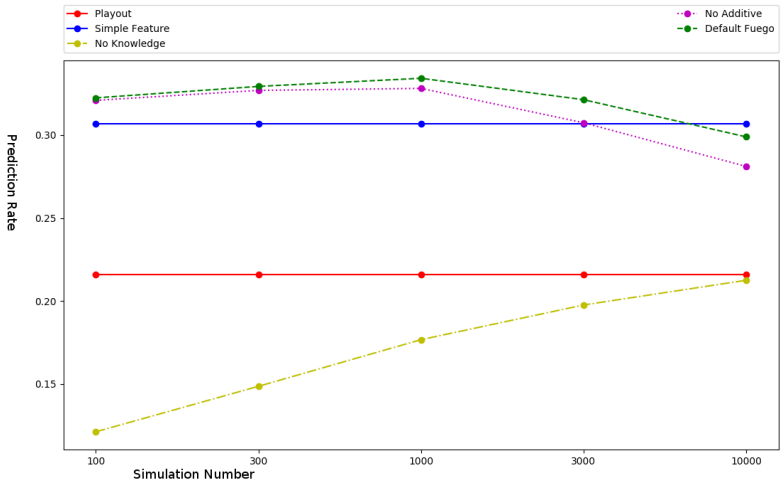


Fig. 1: Graph of move prediction rate.

Removing all knowledge has a big negative impact on the prediction rate in MCTS. It drops the prediction rate to 12% in the No Knowledge player with 100 simulations. Search compensates for the lack of knowledge to some degree. With 10000 simulations, the prediction rate of the No Knowledge player increases to over 21%. Nonetheless, this is still far below the move prediction rate of any MCTS player utilizing knowledge. This shows the role of knowledge in giving direction to MCTS toward nodes with better outlook, when the number of simulations is limited.

The prediction rate of the No Additive player is between approximately 28% and 33%. Up to 1000 simulations, increasing the number of simulations improves the prediction rate; however, after that it starts to drop. When we compare the results of a No Additive player to the Default MCTS-based player with the same number of simulations, we observe a similar pattern in change of prediction rate. The difference between prediction rates of the Default Fuego player and the No Additive player for simulations between 100 to 10000 are: 0.0015, 0.0024, 0.0061, 0.0139, 0.0178. This shows that as the number of simulations grows, additive knowledge slows down the drop of prediction rate in the Default Fuego player, and biases the selection policy in MCTS more towards professional player moves. This widening gap can also be observed in Figure 1.

Given the obtained results several new questions arise:

- Is there any difference in strength between players with similar prediction rate?
- What role does the number of simulations play in players strength vs prediction rate?
- Why does the prediction rate for No Additive and Default MCTS players start to drop?

In order to start investigating these questions, we conducted two experiments. The first experiment measures the playing strength of players against each other. The second measures the move prediction rate in different stages of the games.

5.3 Playing Strength

In order to answer the first two questions in Section 5.2, we created a round robin tournament between all the 11 players described in Section 5.1. Each round consists of 100 games between two players, with each player playing Black 50 times. All players except the Simple Features-Only player use randomization, which resulted in not having any duplicated games. We used GoGui [2] to perform the tournament. We selected some interesting results from this tournament and reported those in Figure 2.

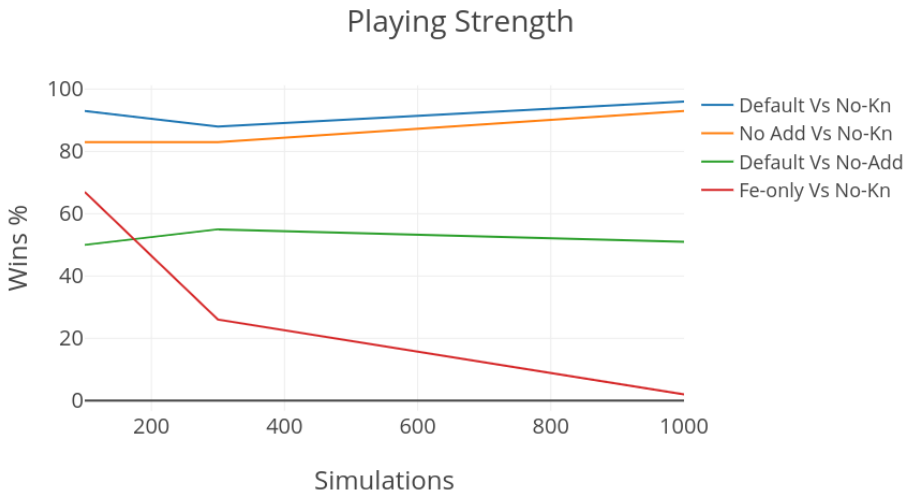


Fig. 2: Result of 100 game matches between pairs of players.

Default MCTS-based Fuego vs No Additive Player This compares the experiments with same number of simulations between the No Additive and default MCTS-based Fuego player. Increasing simulations does not change the balance of strength between these two settings, and removing additive knowledge had minimal impact on playing strength. This is consistent with what we observed in the move prediction task. It can be concluded from the result that these two players have almost the same playing strength against each other when using the same number of simulations.

No Knowledge vs Other MCTS-based Players The playing strength of the No Knowledge player decreases most of the time against an opponent with the same number of simulations as the number of simulations increases. The role of knowledge becomes more important as a player's strength increases. Knowledge helps a player to avoid crucial mistakes in a game, where a stronger opponent can better exploit those mistakes. While it seems that more search should compensate for lack of knowledge, there are two reasons that we do not see that effect in this group of experiments. First, the opponent also benefits from an increased number of simulations. Second, in a player that uses the knowledge, increasing the number of simulations leads to more visits of promising moves that the knowledge picks. This enables the player to examine these moves more deeply, and pick the best among them. The No Knowledge player is less focused and needs more simulations to achieve the same effect.

Varying Number of Simulations, 300 vs 100 As expected from previous experience with MCTS-based engines, we can see that in every case, a 3x increase in number of simulations leads to a huge difference in playing strength. This is in sharp contrast to the move prediction task in Table 1, where the difference was small and sometimes even negative. This shows that using the move prediction rate as a measure to examine a player is not as informative as we expected it to be. There remain aspects of a player which strongly affect its comparative strength against another player, which move prediction is unable to reveal.

No Additive vs Other MCTS-based Players In these experiments, removing the additive term has limited impact on playing strength. The biggest change in playing strength between the No Additive and Default Fuego player is in 300 simulations, where Default Fuego player won 55% of games. Removing feature knowledge decreases the playing strength by a huge margin, with win-rates of only 7-17% for the No Knowledge player.

Simple Features-Only vs No Knowledge Players This scaling experiment shows how much search is needed to reach and surpass Simple Feature knowledge. With 100 simulations, the No Knowledge player is weaker than feature knowledge: it loses 67 games. With 300 simulations, the No knowledge player surpasses the strength of the Simple Features-Only player, and with 1000 simulations the No Knowledge player is much stronger, winning 98 of 100 games.

5.4 A Closer Look at Move Prediction Rate

In Section 5.2 of the previous experiment, surprisingly the move prediction rate did not show any major difference between Default Fuego and the No additive player when the number of simulations was varied between 100 to 1000, while Section 5.3 showed undeniable differences in strength between those players. We also want to understand why the prediction rate starts to drop after 3000 simulations in the Default MCTS-based and No Additive players. In this experiment,

we study the effect of the game phase. We divide a game into six intervals from the opening to the endgame, and measure the prediction accuracy of each player separately for each interval. We created six intervals of 50 moves each, corresponding to move 0 to move 300. Because of the limited number of available samples after move 300 we ignored those final small endgame moves.

Figure 3 shows the move prediction accuracy per interval for Default Fuego with 100 and 1000 simulations, and for No Additive with 100 and 1000 simulations. While Table 1 showed no noticeable difference between 100 and 1000 simulations, Figure 3 shows that for the first 200 moves there is a major difference in both Default Fuego and No Additive players, with a higher prediction rate for the 1000 simulation player. This difference fades from moves $200 - 250$ and turns to the opposite for moves $251 - 300$.

Figure 4 shows the prediction accuracy for experiments where we saw the drop of prediction rate with 3000 and 10000 simulations for No Additive and Default Fuego. We added the 300 simulation players as a baseline. In the opening, the prediction rate for the Default Fuego players increases with number of simulations, and for No Additive players remains very similar for the first 50 moves. From the second interval to the last, the prediction rate of the 300 simulation players sharply increases. For the 3000 simulation players this increase is more moderate. In the 10000 simulation players we observe a drop of prediction rates for the first 250 moves, and then a slight rebound.

To explain the lower prediction rate in the late endgame in players using more simulations, we need to look at how the selection policy in MCTS works. In a game when one player's winning probability is very high, there are many moves that still result in that player winning, while being sub-optimal in terms of score. The selection policy in Fuego maximizes winning probability, not score. After 200 moves, the winner of most of the games can be predicted with high confidence. Fuego chooses a "safest" move according to its noisy simulations. Professional players will not usually select such point-losing "safe" moves. Another reason lies in the impact of knowledge on players with fewer simulations. Knowledge is used to initialize the value of a node in the Monte Carlo tree. When the number of simulations is still small, this initialization plays a major role in MCTS search. Since it is based on features learned from professional games, it biases the search toward professional moves. As the number of simulations grows, its impact diminishes.

5.5 Move Prediction and Feature Frequency

Since the move prediction rate alone does not explain the difference in playing strength, we try to find other differentiating factors between various players by focusing on features. Features play a major role in the success of a player. Even modern neural networks can be seen as a function that is built upon a complex set of features computed in its nodes. In order to understand the significance of different features, we use frequency of features, and we report the most frequent features for each experiment. We count the number of times each feature is present in professional players' moves throughout the game records to identify

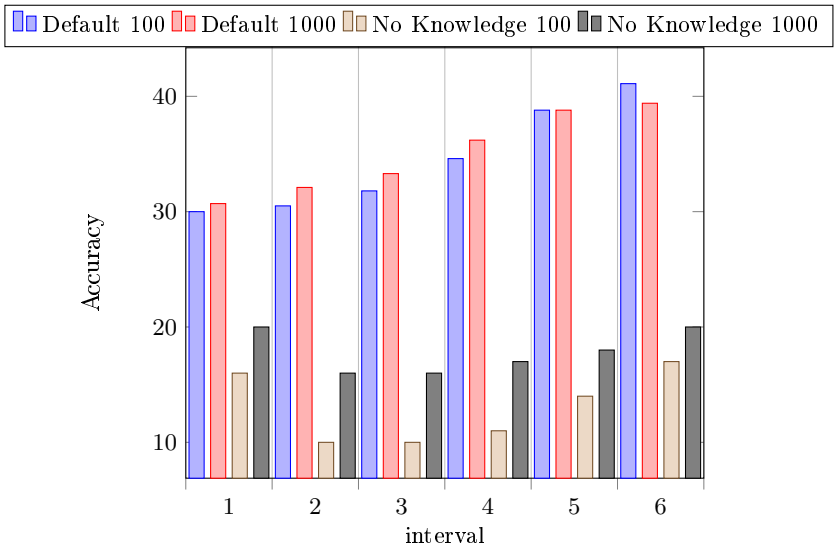


Fig. 3: Move prediction accuracy per game phase for 100 and 1000 simulation players. Each group has 50 moves.

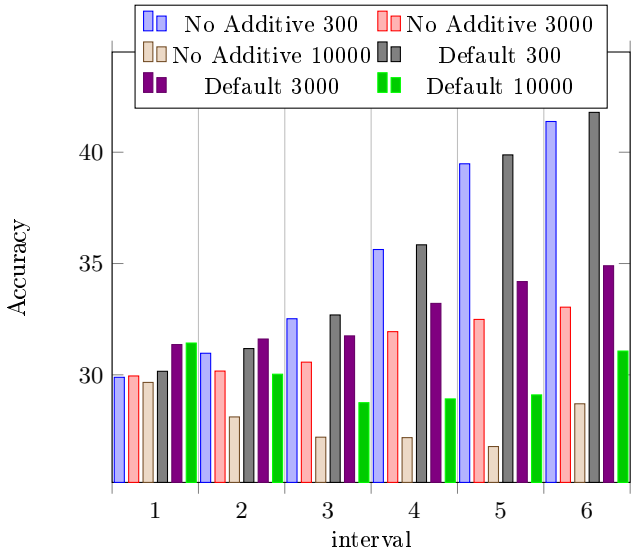


Fig. 4: Move prediction accuracy per game phase for 300, 3000 ad 10000 simulation players. Each group has 50 moves.

frequency of features. We also record the same features over the moves generated by our computer-based players. Our goal is to gain insight on how players differ. In each experiment, we count the number of times each feature exists in the moves generated by one player. The result is a table of features with their significance for the move prediction task.

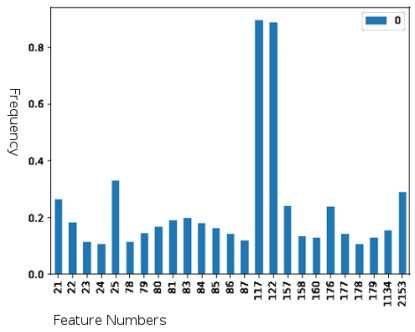
We selected three baselines to analyze the results obtained from our comparison. The first baseline is the frequency of features in all legal moves for every position in all the professional games. The second one is the frequency of all features in professional moves throughout all game records. The last baseline is the frequency of features in the professional move which do not get any attention from our player. In order to determine these moves, we record the number of simulations allocated by Default Fuego for each professional move in each game position. If the number of simulation for the professional move is less than 1% of the move chosen by Default Fuego, that move is marked as a professional move low with simulation count and its features are recorded. Figure 5 shows the graphs for these baselines.

The two most prominent features in Figure 5a are 117 and 122. They represent a distance of 4 or more to the block of the last opponent stone and to the block of the last own stone respectively. Their frequency is more than 85% over all legal moves for each position. This is not surprising due to the size of the 19×19 board, and the distribution of legal moves in each position. The next two prominent features are 25 (moves on line 5 and upward) and 21 (moves on the first line). While moves on line 5 and upward cover 1.68 times the area of moves on the first line, they only happen 1.25 times more in the legal moves. Comparing the frequency of these two features reveals that positions on the first line of the board remain empty longer than other points in professional games.

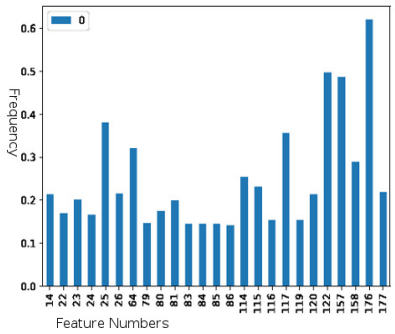
Figure 5b shows features of professional players moves. Feature 176 (distance 2 to closest opponent stone) is true for 62% of professional moves and feature 177 (distance 3 to closest opponent stone) in 22%. In total 85% of professional moves are in close proximity to opponent stones. Feature 157 and 158 (distance 2 and 3 to closest own stone) together cover almost 80% of professional moves, showing that professionals play close to their own stones as well.

As in Figure 5a, in Figure 5c prominent features of professional moves missed by Fuego are 122 and 117 with frequency of 68% and 60%. This shows that moves that usually get ignored by Fuego are non-local responses to the opponent, or "tenuki" moves that change the area of play.

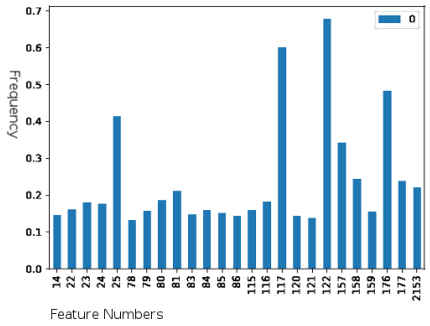
Impact of More Simulations Figure 6 shows the difference in feature frequency of moves generated by default Fuego with 3000 and 100 simulations. The main difference is in features 117 and 122 which indicate changing the area of play, "tenuki". Feature 25 (play on line number 5 and up) is another example of the impact of more simulations on the area of play. We saw that this is one of the prominent features of professional players moves. These results show that the player with more simulations can find centre and tenuki moves more often, and becomes more similar to how professional players play in these situations.



(a) Feature frequencies of every legal move in professional games.



(b) Feature frequencies of all professional moves in professional games.



(c) Features of professional moves that have low number of simulations compared to the move played by Default Fuego using 1000 simulations.

Fig. 5: Feature counts of baselines.

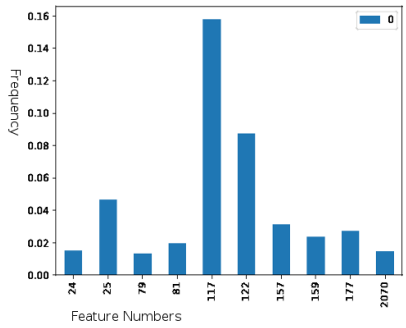


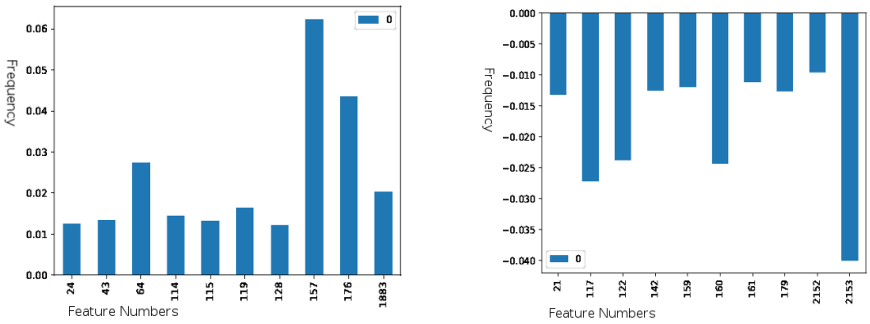
Fig. 6: Top 10 differences between features count of default Fuego player with 3000 simulations and 100 simulations.

Impact of the Additive Term Figures 7a and 7b show the differences between the default Fuego player and the No additive player with 3000 simulations. In Figure 7a, features 157 and 176 are for playing in distance of 2 to the closest own stone and opponent colour respectively. They happen 6% and 4% more in the default Fuego player which benefits from the additive term. This shows that additive knowledge encourages playing close to previous stones. Feature 64 also happens 3% more in the player using the additive term. This feature is for 3×3 patterns used in the simulations policy. This is an expected behaviour as the additive knowledge uses a local shape pattern to evaluate each move. Other features in Figure 7a have a very low frequency.

The No Additive player plays more often in empty areas of the board (feature 2153, 3×3 empty pattern), and far from all other stones, features 117, 122 and 160 (distance 5 to closest own stone).

Impact of Simple Feature Knowledge with Increasing Number of Simulations By comparing Figures 8a and 8b and Figures 8c and 8d we can understand the impact of simple feature knowledge. Features 26 (distance 2 to last opponent stone), 64 and 114 (distance 1 to block of last opponent stone) are more present in the player with knowledge, while in Figure 8b features 117 and 122 occur up to 42% more in the No Knowledge player. This shows that the No Knowledge player with low number of simulations plays more randomly in all areas of the board without any attention to the last own or opponent move, while the player with knowledge responds locally to those moves more often.

As the number of simulations grows, we still observe in Figures 8c and 8d the same difference in style of play from default Fuego and the No Knowledge player. This gap, however, narrows to half with consistency in relative frequency of features to each other. To some degree more simulations compensate for the lack of knowledge in the No Knowledge player, as we already observed in the



(a) Top 10 positive differences.

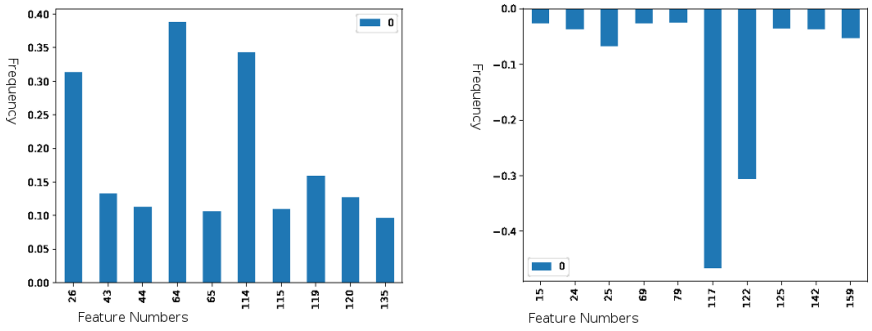
(b) Top 10 negative differences.

Fig. 7: Difference between feature counts of default Fuego and No Additive player when both players use 3000 simulations.

move prediction task; however, more simulations are not able to completely close the gap.

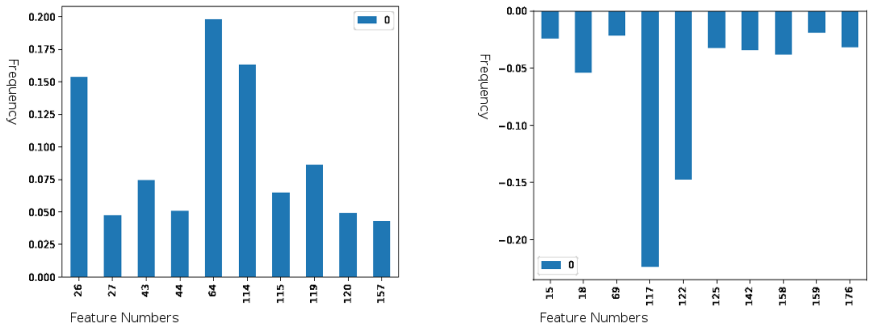
Features of Professional Moves We ran another experiment to understand why some professional moves are ignored in Fuego. We compared the statistics of the default Fuego moves to the professional moves with low simulations in Figure 9. This helps to understand what kind of moves professional players make that Fuego does not consider, and how often those moves happen. In Figure 9a, features 114, 115, 119, 157, 176 are all for moves with distance of 1 or 2 to the own or opponent stones. This signifies the higher degree of locality of play in Fuego versus professional players. Also 3×3 simulation policy patterns (feature 64) occur 35% more in the default Fuego moves than in professional moves with low number of simulations, showing that many professional moves do not follow traditional 3×3 patterns as described in [11]. Looking at Figure 9b, features 117, 122, 159, 160, 161, 178, 179 are all for moves with distance of 4 or more to stones of either colour and feature 2153 is for the empty 3×3 square. These features happen up to 24% more in professional moves that received a very low number of simulations from Fuego. This shows that Fuego systematically likes to play locally, and moves with longer distance to the last own or opponent stone are not appealing to the program.

Feature differences in Figures 9c and 9d between the default Fuego moves and all professional moves have similar feature differences to Figures 9a and 9b, but with different magnitude. First the magnitude of difference is much lower in Figures 9c and 9d. The other difference is that the most differentiating factor for the default Fuego player is that it plays 12% more in distance 2 of opponent stones (feature 176) than professional players. Professional moves still occur more in distance of 3 or more (features 116, 159, 160, 177, 178 and 179) to other stones, but the gap to Fuego is smaller.



(a) Top 10 positive difference when both players use 100 simulations.

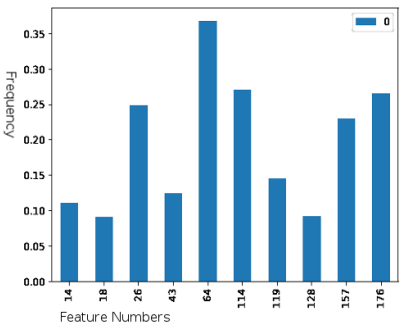
(b) Bottom 10 negative difference when both players use 100 simulations.



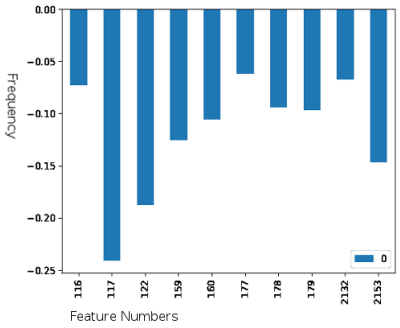
(c) Top 10 positive difference when both players use 3000 simulations.

(d) Bottom 10 negative difference when both players use 3000 simulations.

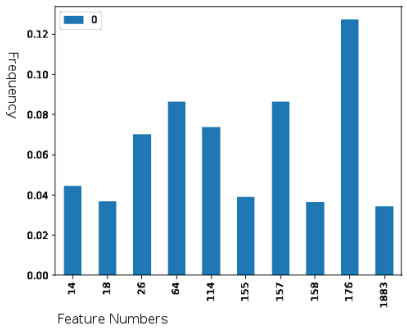
Fig. 8: Difference between feature counts of default Fuego and No Knowledge player.



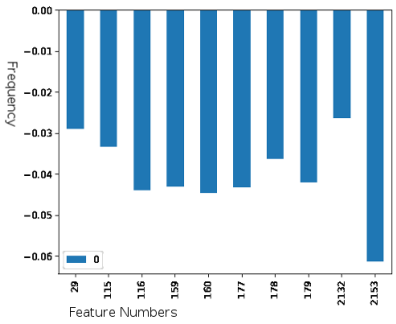
(a) Positive difference with professional moves low simulation count.



(b) Negative difference with professional moves with low simulation count.



(c) Positive difference with all professional moves.



(d) Negative difference with all professional moves.

Fig. 9: Difference between feature counts of default Fuego with 3000 simulations moves and professional moves.

5.6 Move Selection Analysis

The next experiment helps us to understand under what circumstances a player can predict a professional move, while at other times it can not. We created an experiment to measure the number of simulations relative to the initial weight of a move. The results of this experiment are reported in Figure 10.

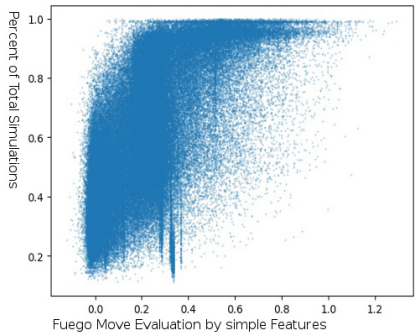
For the Y-axis of Figure 10 we measured two different cases. In the first case, we measured the number of simulations $sim_{s,a}$ for move a in state s relative to the total number of simulations for state s in the professional game: $\frac{sim_{s,a}}{\Sigma_i sim_{s,i}}$. For the second case, we measured the relative number of simulations $sim_{s,a}$ for move a in state s to the number of simulation $sim_{s,b}$ for move b in state s: $\frac{sim_{s,a}}{sim_{s,b}}$. The Y-axis of Figures 10a to 10d use the first case. For Figures 10a and 10b, move a is the move selected by default Fuego, and for Figures 10c and 10d it is the move selected by the professional player. The Y-axis of Figures 10e and 10f uses the second case. Move a is the move selected by the professional player and move b is the move selected by default Fuego.

The X-axis of Figure 10 has two different formats. In the first one, we use the initial weight $w_{s,a}$ of move a in state s of the professional game. For the second case, we compute the maximum weight $w_{s,max}$ for the state s, then compute the relative weight of move a to the maximum weight $\frac{w_{s,a}}{w_{s,max}}$. Since the weight of a move can be negative, we normalize the relative value by a sigmoid function $\frac{sig(w_a)}{sig(w_{max})}$. The X-axis of Figures 10a, 10c and 10e uses the first format. The X-axis of Figures 10b, 10d and 10f uses the second format. Move a is selected by default Fuego in Figures 10a and 10b, and by professional players in Figures 10c to 10f.

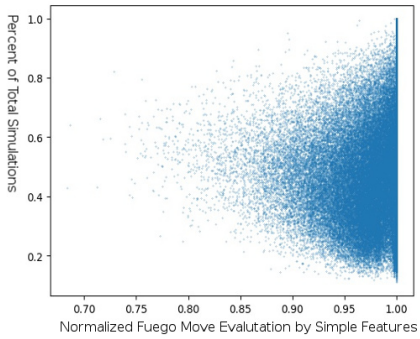
In order to understand the distribution of simulations, we created Figure 11a. It represents the relation between the weight of the feature for a move selected by default Fuego and the percent of simulations that move has received. Most of the moves selected by default Fuego have the majority of the simulations. Moves with higher initial weights receive almost 100% of simulations. Moves selected by Fuego have different ranges of weights from low to high. However, Figure 10b shows that even moves with low weights have weights close to the maximum weight of that position, and most of the times are the maximum weight.

Figure 10c shows that professional players moves most of the time either received the maximum number of simulations, or received close to zero. Moves that have an in-between number of simulations make up a smaller portion of professional moves. Figure 10d better illustrates this point. Figure 11b shows that for professional moves to get the attention of Fuego, they need to have higher evaluation by simple features.

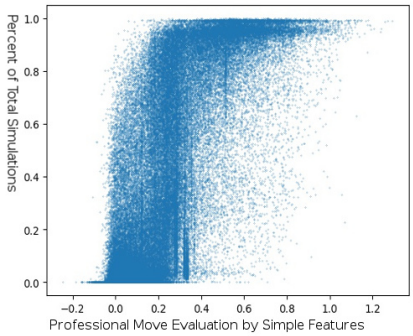
We also compared the number of simulations for the professional moves and the moves selected by default Fuego. Figure 10e shows that very often the move played by professionals is the same as the Fuego move. However, if they differ, the chances of the professional move having a large number of simulations is low. Most of the time, it has less than 20% compared to Fuego's move. Figure 10f plots the relative number of simulations and the relative heuristic weight of the



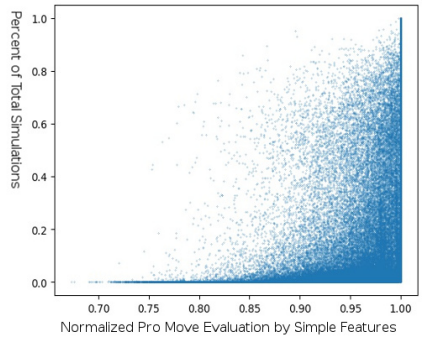
(a) Graph of selected move by default Fuego player given its initial weight



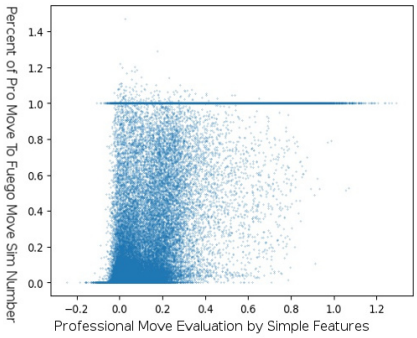
(b) Graph of selected move by default Fuego player given sigmoid of its initial weight divided by sigmoid of Max weight.



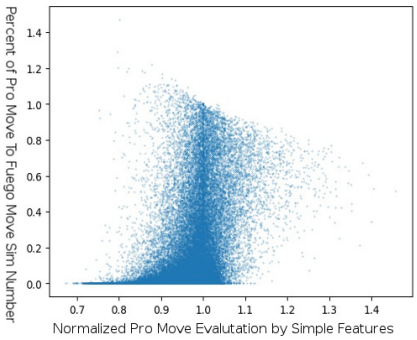
(c) Graph of played move by professional player given its initial weight.



(d) Graph of selected move by professional Player given its sigmoid of its initial weight divided by sigmoid of Max weight.

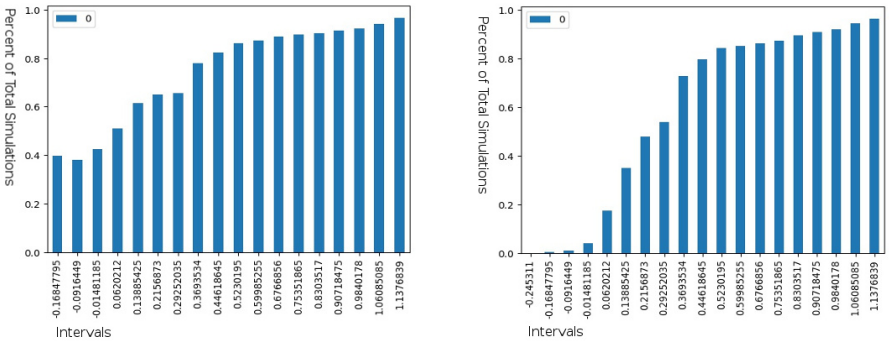


(e) Percent of simulations relative to selected move divided by professional player's move given its initial weight.



(f) Percent of simulations relative to selected move divided by professional player's move given sigmoid of its initial weight divided by sigmoid of Max weight.

Fig. 10: Comparison between number of simulations for initial feature weight.



(a) Graph of average simulations for selected move by default Fuego player given its initial weight.

(b) Graph of average simulations for selected move by default professional player given its initial weight.

Fig. 11: Percent of average number of simulations for buckets of initial weights.

professional move to the move selected by default Fuego. The ratio of simulations drops sharply as the relative weight of the professional player's move decreases. For professional moves that have a ratio of less than 0.9, their number of simulations is near zero most of the time. Less than 7% of professional moves have both higher weight than the move selected by Fuego, and fewer simulations.

This experiment showed us the importance of simple feature initialization on the number of simulations a move receives. Fuego gives professional moves more simulations if they have high evaluation by simple features and ignores them if their simple feature evaluation is low.

The move selected by Fuego does not need to have high evaluation as seen in Figure 10a. It just needs to have an evaluation close to the maximum move evaluation of that position. This can be observed in Figure 10b. We also observed in Figures 10d and 10e how professional moves either receive close to the maximum number of simulations or close to zero.

6 Conclusions and Future Work

In this work we investigated two popular evaluation methods: move prediction and playing strength, and how they relate to each other. We noticed that move prediction did not reveal important aspects of a player, and there remain many details that an aggregated move prediction percentage can not express. Players with similar move prediction rate can have very different playing strengths.

We used a playing strength experiment to understand the impact of the following concepts in MCTS: additive knowledge, simple feature knowledge, number of simulations, and playout policy. The additive term has a very small impact

on playing strength. Removing feature knowledge has a massive negative impact on playing strength which only increases with more search.

We analyzed the move prediction rate in several game stages in order to capture differences between the players. With more search, the move prediction rate drops near the end of a game, due to "safe" move selection in MCTS.

To find more differentiating factors between players, we examined feature frequencies in the move prediction task for different players. We were able to find features that differ remarkably between players, which can be used to define their behaviour. We also found relations between the evaluation of feature knowledge and the number of simulations a move receives.

For future work, we want to extend the study by including neural network-based players and extending the experiments to understand the impacts of a neural network in detail. Another promising extension of this work is trying to understand neural networks in terms of both simple features and move prediction, in order to find an interpretation of their behaviour with known features of the Go game.

7 Acknowledgement

Financial support was provided by NSERC, The Natural Sciences and Engineering Research Council of Canada.

References

1. Fuego Source Code, `http://fuego.sourceforge.net`, SVN revision 2032, updated on Aug 16, 2016
2. GoGui Project, `https://sourceforge.net/projects/gogui/`, Accessed Online on Dec 14, 2016
3. Professional Games, `https://badukmovies.com/pro_games`, Accessed Online on Nov 02, 2016
4. Cazenave, T.: Residual networks for computer Go. IEEE Transactions on Games **10**(1), 107–110 (2018)
5. Clark, C., Storkey, A.: Training deep convolutional neural networks to play Go. In: International Conference on Machine Learning. pp. 1766–1774 (2015)
6. Coulom, R.: Computing Elo Ratings of Move Patterns in the Game of Go. In: van den Herik, H.J., Winands, M., Uiterwijk, J., Schadd, M. (eds.) Computer Games Workshop. Amsterdam, Netherlands (Jun 2007)
7. Elo, A.E.: The rating of chessplayers, past and present. Arco Pub. (1978)
8. Enzenberger, M., Müller, M., Arneson, B., Segal, R.: Fuego an open-source framework for board games and Go engine based on Monte Carlo tree search. IEEE Transactions on Computational Intelligence and AI in Games **2**(4), 259–270 (2010)
9. Fernando, S., Müller, M.: Analyzing Simulations in Monte Carlo Tree Search for the Game of Go. In: Computers and Games - 8th International Conference, CG 2013, Yokohama, Japan, August 13-15, 2013, Revised Selected Papers. pp. 72–83
10. Gelly, S., Silver, D.: Combining online and offline knowledge in UCT. In: Proceedings of the 24th international conference on Machine learning. pp. 273–280. ACM (2007)

11. Gelly, S., Wang, Y., Munos, R., Teytaud, O.: Modification of UCT with patterns in Monte Carlo Go (2006)
12. Haqiqat, F.: Analyzing the impact of knowledge and search in Monte Carlo Tree Search in Go. Master's thesis, University of Alberta (2018)
13. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
14. Huang, S.C., Coulom, R., Lin, S.S.: Monte Carlo Simulation Balancing in Practice. In: Proceedings of the 7th International Conference on Computers and Games. pp. 81–92. CG'10, Springer-Verlag, Berlin, Heidelberg (2011)
15. Maddison, C.J., Huang, A., Sutskever, I., Silver, D.: Move evaluation in Go using deep convolutional neural networks. International Conference on Learning Representations. arXiv preprint arXiv:1412.6564 (2014)
16. Martin Wistuba and Lars Schmidt-Thieme: Move Prediction in Go – Modelling Feature Interactions Using Latent Factors. In: KI 2013: Advances in Artificial Intelligence, pp. 260–271. Springer Berlin Heidelberg (2013)
17. Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., et al.: Mastering the game of Go with deep neural networks and tree search. Nature **529**(7587), 484–489 (2016)
18. Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al.: Mastering the game of Go without human knowledge. Nature **550**(7676), 354–359 (2017)
19. Silver, D., Tesauro, G.: Monte Carlo Simulation Balancing. In: Proceedings of the 26th Annual International Conference on Machine Learning. pp. 945–952. ICML '09, ACM, New York, NY, USA (2009)
20. Xiao, C., Müller, M.: Factorization Ranking Model for Move Prediction in the Game of Go. In: AAAI. pp. 1359–1365 (2016)
21. Xiao, C., Müller, M.: Integrating Factorization Ranked Features in MCTS: An Experimental Study. In: Computer Games, pp. 34–43. Springer (2016)

A Detailed Move Prediction Results

Experiment	Accuracy
Playout Policy-Only	0.2160
Simple Features-Only	0.3066
No Knowledge 100	0.1212
No Knowledge 300	0.1486
No Knowledge 1000	0.1767
No Knowledge 3000	0.1976
No Knowledge 10000	0.2125
No Additive 100	0.3209
No Additive 300	0.3269
No Additive 1000	0.3281
No Additive 3000	0.3074
No Additive 10000	0.2811
Default 100	0.3224
Default 300	0.3293
Default 1000	0.3342
Default 3000	0.3213
Default 10000	0.2989

Table 1: Result of move prediction for players based on Fuego.