THE APPLICATION OF FUZZY REASONING TO THE OPENING GAMES OF 19×19 GO

Byung-Doo Lee
Dept. of Computer Science
Univ. of Auckland
Auckland, New Zealand
blee026@cs.auckland.ac.nz

Hans Werner Guesgen
Dept. of Computer Science
Univ. of Auckland
Auckland, New Zealand
hans@cs.auckland.ac.nz

Jacky Baltes
Dept. of Computer Science
Univ. of Manitoba
Winnipeg, Canada
jacky@cs.umanitoba.ca

Soo-Hyun Jeong
Dept. of Baduk Studies
Univ. of Myongji
Kyunggido, Korea
shjeong@mju.ac.kr

ABSTRACT
This paper describes the result of applying a fuzzy reasoning method, which conducts Go term knowledge based on pattern knowledge, to the opening game in Go. We discuss the implementation of the fuzzy reasoning method for deciding the best next move to proceed through the opening game. We also let the fuzzy reasoning method play against the TD(λ) learning method to compare the performance. The results reveal that the simple fuzzy reasoning system performs better than the TD learning method and it shows great potential to be applied to the real game of Go.

KEY WORDS
Computer Go, Fuzzy Reasoning, Temporal Difference Learning, Neural Network, Distance Transform

1 Introduction

Go is a board game for two players: black and white. Two players alternately play a single stone onto the 19×19 intersections (i.e. 361 points). The goal in Go is to surround more territory than the opponent by enclosing some area.

Since the Go board is mostly empty and a branching factor is over 200 [1] in the early stages of the opening game, $\alpha\beta$ search or other more advanced search algorithms are not tractable. Furthermore, the number of moves is approximately between 250 and 300. This means that any naive search method is doomed to failure. In fact, the search spaces need to be strongly biased so that efficient problem solving strategies can be developed. One such approach is the use of fuzzy reasoning which is an inference procedure to derive a conclusion using a set of fuzzy if-$then$ rules.

Human players intuitively play a game with both low level concepts (e.g., counting the number of liberties of a string) and high level concepts (e.g., the safety of a group, connectivity between groups, or thickness of a group). Since most of accumulated Go knowledge is implicitly in the form of patterns, it is easy for humans to perceive it but hard to model it with a computer.

Since Go knowledge is composed of pattern knowledge and term knowledge [1], the strong points when they complement each other are as follows:

- Pattern knowledge provides a sequence of moves (or a move) for tackling board situations, since expert games (patterns) implicitly involve most Go knowledge [2].
- Term knowledge, which can conduct high-level concepts and complements the weakness of pattern knowledge, provides a move for tackling unrecognised patterns, since there is a very low possibility that an identical pattern will ever be encountered more than once.

For solving opening games of Go, fuzzy logic, which is a rule-based method for solving problems in an uncertain (imprecise) domain, was applied. This approach aims at deciding the best next move with term knowledge based on pattern knowledge.

2 Fuzzy Inference System

A fuzzy inference system (FIS) is a nonlinear mapping of crisp inputs into crisp outputs using fuzzy rules, and is composed of three parts: fuzzification, fuzzy inference and defuzzification.

The Mamdani fuzzy model uses the minimum (for T-norm) and the maximum (for T-conorm) operators for fuzzy relational composition. Assume that there are n Single-Input-Single-Output (SISO) fuzzy rules to be fired. Then the i-th rule is represented as:

$$R_i: \quad \text{if } x \text{ is } A_i \quad \text{then } y \text{ is } B_i \tag{1}$$

where i = 1, 2, $\cdots$ and n.

The linguistic variables x and y are input and output fuzzy variables, respectively. Fuzzy sets A_i and B_i are linguistic terms represented as membership functions.

Then, the fuzzy relation of the i-th rule R_i on $X \times Y$ is expressed by $A_i \rightarrow B_i$ as:

$$\mu_{R_i}(x, y) = \min\left[\mu_{A_i}(x), \mu_{B_i}(y)\right] = \mu_{A_i}(x) \wedge \mu_{B_i}(y) \tag{2}$$

where the minimum ($\wedge$) operator is applied on the Cartesian product space of X and Y.

Finally, the fuzzy relation R (the aggregated relation, representing the entire fuzzy model) is derived by summing

over all of the fuzzy relations as:

$$R = \bigcup_{i=1}^{n} R_i = \bigcup_{i=1}^{n} \mu_R(x,y) = \bigcup_{i=1}^{n} [\mu_{A_i}(x) \wedge \mu_{B_i}(y)]. \quad (3)$$

If the input fuzzy set $x = A'$ is given, the output fuzzy set B' is computed based on the max-min composition by:

$$\begin{aligned} \mu_{B'}(y) &= \vee_x [\mu_{A'}(x) \wedge \mu_R(x,y)] \\ &= \bigcup_{i=1}^{n} \{\vee_x [\mu_{A'}(x) \wedge \mu_{A_i}(x)] \wedge \mu_{B_i}(y)]\} \\ &= \bigcup_{i=1}^{n} [\alpha_i \wedge \mu_{B_i}(y)] \end{aligned} \quad (4)$$

where α_i is the *firing strength*.

For extracting a crisp output value from a fuzzy set B', we need to defuzzify the fuzzy set B'. As a defuzzification method, the centre-of-gravity (COG) method is mostly used and is defined as:

$$Y_{COG} = \frac{\int_Y \mu_{B'}(y)\ y\ dy}{\int_Y \mu_{B'}(y)\ dy} \quad (5)$$

where $\mu_{B'}(y)$ is the aggregated output membership function, representing a relationship of the fuzzy model.

3 Implementation

The fuzzy reasoning system in Figure 1 has three basic components: a neural network with supervised learning, an evaluator of board situation with each candidate next move, and a neuro-fuzzy controller with the Sugeno fuzzy inference model [3]. The fuzzy reasoning system performs the following three steps:

- Ten candidate moves are generated by the neural network with supervised learning, which can recognise patterns of the prototypical opening games.
- Each candidate move is evaluated by a Distance Transform (DT) method [4] to provide two crisp inputs to the neuro-fuzzy controller: the change rate of the net influence and the change rate of the potential net territory.
- The neuro-fuzzy controller decides the best move among the candidate moves by the fuzzy inference process.

3.1 Network with Supervised Learning

In an opening game of Go, most human players select the best next move from similar patterns which they have remembered. That is, humans recognise similar features from previously played games and prototypical games.

We constructed a back-propagation neural network with supervised learning to recognise the patterns of the prototypical opening games. The main reasons for constructing a neural network are (1) to mimic human behaviour of remembering the patterns (games) played and (2) to reduce computing time.

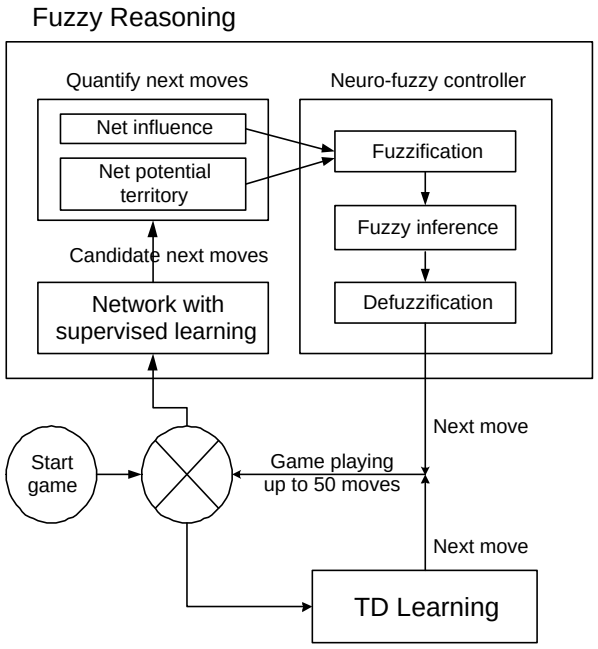


Figure 1. A scheme to compete a fuzzy reasoning system with a Temporal Difference (TD) learning.

For training the network, the encoded board positions of each opening game were fed into a single-layered neural network (i.e. a 362-361 net) with the tangent hyperbolic function as an activation function. The input data is a sequence of the form $< S_0, S_1, ..., S_f >$ for an opening game, where S_0 denotes an empty (initial) board situation and f indexes the board situation after the final move of an opening game (e.g., f = 47 in Figure 3(a)).

Given the delta rule, the prototype supervised learning procedure with the hyperbolic tangent activation function is:

$$\Delta w_{ji} = \kappa(z_j - P_j)\frac{\partial P_j}{\partial w_{ji}} = \alpha(z_j - P_j)(1+P_j)(1-P_j)X_i \quad (6)$$

where α is the learning rate, z_j is the j-th target value (which has a value of +1, 0 or -1) and P_j is the j-th output generated by the network. X_i is the i-th component of pattern S_t at time t.

We created a database of 213 professional opening games manually. The games were selected from [5]. We then generated 1,704 games by using the symmetries (i.e. reflection and rotation) of each game.

Figure 2 shows a sequence of moves generated by the trained neural network, and the performance of predicting next moves is surprisingly good. But the trained network has some problems when being applied to a real game: i.e. it sometimes generates strange moves such as

black 47 in Figure 2 because of (1) a lack of understanding the importance of the temporal sequence, and (2) insufficient Go knowledge for tactical and strategic playing. These weak points can be compensated by using a fuzzy reasoning method embedded in Go knowledge.

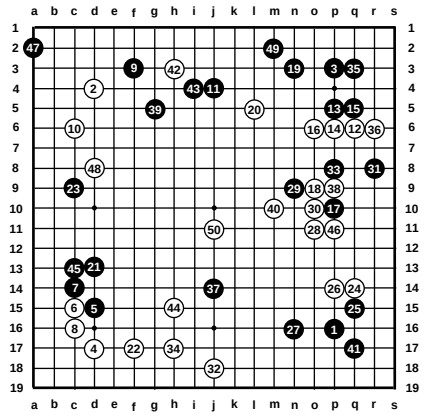


Figure 2. A sequence of moves generated by the trained neural network after 100 epochs. The number of moves in an opening game was limited to 50.

For reducing computing time (i.e. playing a move in a limited computing time), we therefore generated a set of candidate next moves ordered (prioritised) by the activation function output values. Among them we tried to pick the best one as a next move through the fuzzy rule-based inference system.

3.2 Evaluation of Candidate Moves

There is no simple theory which determines the best next move in an opening game. However, there are some popular general principles that can be helpful. For instance:

- Corners, sides and then center.
- Play in the center of a symmetrical formation.

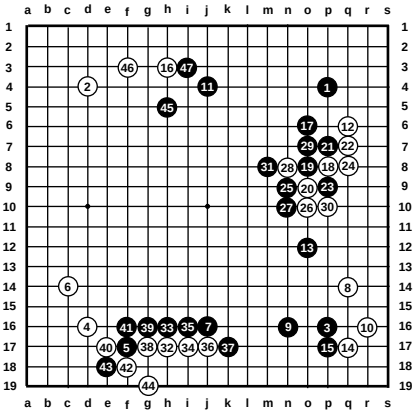
Linguistic expressions such as these are so vague that it is very difficult to implement Go knowledge into a computer system. The aim of the opening game is to play stones efficiently to gain influence (rather than to make territory directly) from which a player can develop the territory gradually. Influence provides a high possibility of maximising territory acquired later in the opening game.

Influence, which is strongly related to the Go theory concept of *thickness*, is a simple heuristic for evaluating board positions. Each stone radiates influence on the board. Influence from a stone propagates out and decreases with distance. Captured stones have no influence, and surrounded stones influence only as far as the surrounding group. We applied an influence function that depends on the shortest distance between two points as:

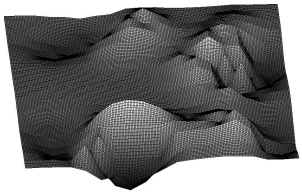
$$I(d) = C_1 \cdot exp(-d^2/4) \quad (7)$$

where d is the shortest empty path between a position influenced and a stone influencing, and C_1 is a constant: +64 for black and -64 for white.

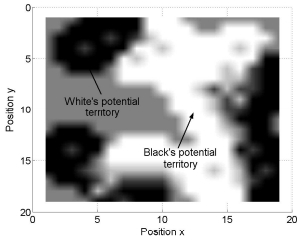
For finding a shortest empty path d between two points, we applied the Distance Transform (DT) method, which is widely used in image processing [4]. This method finds all paths from the goal location (e.g., stones influencing) back to the start location (e.g., a position being influenced). Figure 3(b) illustrates the influence map for black and white of the opening game in Figure 3(a) performed by Equation 7.



(a) A trained opening game



(b) The influence map.



(c) The territory map

Figure 3. (a) An opening game among 1,704 opening games used in training the network. The number on each stone denotes its move sequence. (b) The influence map after applying the influence function. Hills represent black's influence and valleys represent white's influence. (c) The territory map derived from Figure 3(b).

For crisp inputs of the fuzzy controller, we computed the change rate of the net total influence when playing each of the 10 candidate moves. The change rate of the net total influence of the i-th move is defined by:

$$DI(i) = exp(-i^2/2\sigma^2) \cdot (NI(i-1) - NI(i))/C_2 \quad (8)$$

where $DI(i) \in$ [0,1], i = 1, 2, $\cdots$, 10 and the degree of similarity, $exp(-i^2/2\sigma^2)$, measures the priority of each candi-

date move generated by the network (σ = 45).

The net influence $NI(i)$ is the summation of the overall black influence net of the overall white influence when the i-th stone is played on the Go board (C_2 = 1,000).

Another input variable is the change rate of the potential territory as:

$$DT(i) = exp(-i^2/2\sigma^2) \cdot (NT(i-1) - NT(i))/C_3 \quad (9)$$

where $DT(i) \in$ [0,1], i = 1, 2, $\cdots$, 10 and C_3 = 50.

Potential territory $NT(i)$ was calculated from the influence map by truncating extreme highs and lows at the same absolute elevation and excluding the points already played by black and white.

3.3 Fuzzy Controller

A neuro-fuzzy controller is generally modelled from the three most commonly used fuzzy inference systems (FIS): the Mamdani FIS, the Tsukamoto FIS and the Sugeno FIS [6]. From [7], the Sugeno neuro-fuzzy model is far superior to the Mamdani neuro-fuzzy model (i.e. in terms of lower RMSE error and less memory) but the implementation of the Sugeno neuro-fuzzy model is more difficult than that of the Mamdani neuro-fuzzy model [8].

We constructed a similar Adaptive Neural Fuzzy Inference System (ANFIS) using the Mamdani neuro-fuzzy model and the Sugeno neuro-fuzzy model as a neuro-fuzzy controller, which is called the Similar ANFIS (SANFIS) from now.

The ANFIS shown in the right side of Figure 4 is a neuro-fuzzy inference system which has Sugeno's type of if-$then$ rules. ANFIS identifies antecedent and consequent parameters by using a supervised neural network with the back-propagation method. That is, ANFIS determines the antecedent parameters using back-propagation learning and the consequent parameters using least-square estimation [9]. We tried to adapt only the consequent parameters (i.e. p_i, q_i and r_i) in SANFIS rather than to adapt the antecedent and the consequent parameters as in ANFIS.

SANFIS qualitatively evaluates the candidate next moves depending on the two fuzzy input variables: the change rate of the net influence, x, and the change rate of the potential territory, y. With two linguistic input variables and an output variable z, we defined the Sugeno neuro-fuzzy controller in three Multi-Input-Single-Output (MISO) fuzzy rules as:

$$\begin{array}{ll} \text{Rule 1:} & \text{if } x \text{ is } HIGH \text{ and } y \text{ is } LARGE \\ & \text{then } z_1 = p_1 x + q_1 y + r_1, \\ \text{Rule 2:} & \text{if } x \text{ is } MIDDLE \text{ and } y \text{ is } MEDIUM \\ & \text{then } z_2 = p_2 x + q_2 y + r_2, \\ \text{Rule 3:} & \text{if } x \text{ is } LOW \text{ and } y \text{ is } SMALL \\ & \text{then } z_3 = p_3 x + q_3 y + r_3. \end{array} \quad (10)$$

When the inputs $x = x_0$ and $y = y_0$ are given, the firing strength of the i-th rule, α_i, is computed by the minimum operator as:

$$\begin{array}{rcl} \alpha_1 & = & HIGH(x_0) \wedge LARGE(y_0), \\ \alpha_2 & = & MIDDLE(x_0) \wedge MEDIUM(y_0), \\ \alpha_3 & = & LOW(x_0) \wedge SMALL(y_0) \end{array} \quad (11)$$

and the outputs of each rule derived from the relationships in Equation 10 are $z_i = p_i\ x_0 + q_i\ y_0 + r_i$, where i = 1, 2 and 3.

Finally, the final crisp output z_0 is computed by the center-of-gravity method as:

$$z_0 = \frac{\alpha_1 z_1 + \alpha_2 z_2 + \alpha_3 z_3}{\alpha_1 + \alpha_2 + \alpha_3} = \beta_1 z_1 + \beta_2 z_2 + \beta_3 z_3 \quad (12)$$

where β_i is the normalised value of α_i with respect to the sum ($\alpha_1 + \alpha_2 + \alpha_3$).

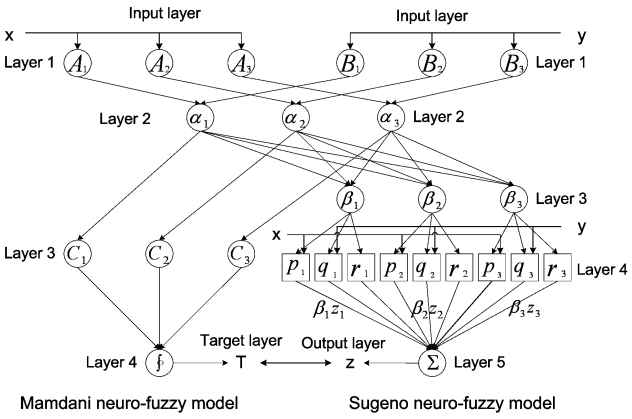


Figure 4. An adaptive network with a Sugeno neuro-fuzzy model and a Mamdani neuro-fuzzy model to obtain the consequence parameters.

The arising problem is how to obtain the consequent parameter values, p_i, q_i and r_i. To obtain these values, we implemented the Mamdani neuro-fuzzy controller in the left side of Figure 4 to generate the crisp output which was used as the target value to train the Sugeno neuro-fuzzy controller. And the rules in Equation 10 were replaced by:

$$\begin{array}{ll} \text{Rule 1:} & \text{if } x \text{ is } HIGH \text{ and } y \text{ is } LARGE \\ & \text{then } z \text{ is EXCELLENT,} \\ \text{Rule 2:} & \text{if } x \text{ is } MIDDLE \text{ and } y \text{ is } MEDIUM \\ & \text{then } z \text{ is MEDIUM,} \\ \text{Rule 3:} & \text{if } x \text{ is } LOW \text{ and } y \text{ is } SMALL \\ & \text{then } z \text{ is POOR.} \end{array} \quad (13)$$

The bell-shaped Gaussian membership function $\mu(t; \sigma, c)$ was used for the memberships of antecedent and consequent parts of the rules as shown in Table 1. The function has the form:

$$\mu(t; \sigma, c) = exp\left(\frac{-(t-c)^2}{2\sigma^2}\right) \quad (14)$$

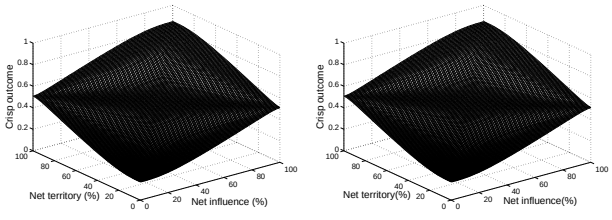
where c represents the membership function's centre and σ determines the membership function's width.

Figure 5(a) illustrates the control surface of the function $z = f(x, y)$, generated by the Mamdani neuro-fuzzy controller.

Part of	Linguistic terms	Applied MFs
Antecedent	$HIGH$	$\mu_{A_1}(x; 0.2, 1.0)$
	$MIDDLE$	$\mu_{A_2}(x; 0.1, 0.5)$
	LOW	$\mu_{A_3}(x; 0.2, 0.0)$
	$LARGE$	$\mu_{B_1}(y; 0.2, 1.0)$
	$MEDIUM$	$\mu_{B_2}(y; 0.1, 0.5)$
	$SMALL$	$\mu_{B_3}(y; 0.2, 0.0)$
Consequent	$EXCELLENT$	$\mu_{C_1}(z; 0.2, 1.0)$
	$MEDIUM$	$\mu_{C_2}(z; 0.1, 0.5)$
	$POOR$	$\mu_{C_3}(z; 0.2, 0.0)$

Table 1. Applied membership functions.

Based on the control surface of the Mamdani neuro-fuzzy controller, the Sugeno neuro-fuzzy controller was trained to obtain the consequent parameters enabling the control of the neuro-fuzzy controller. After 100 epochs, we obtained the consequent parameters of the Sugeno neuro-fuzzy controller. With these fixed consequent parameters, the Sugeno neuro-fuzzy controller generated the control surface illustrated in Figure 5(b).



(a) Control surface 1 (b) Control surface 2

Figure 5. (a) Control surface generated by the Mamdani neuro-fuzzy controller and (b) Control surface generated by the Sugeno neuro-fuzzy controller after training (learning rate α=0.01, epochs=100).

The control surface generated by the Sugeno neuro-fuzzy controller demonstrated that:

- the performance of a supervised neural network in the Sugeno neuro-fuzzy controller is very good (i.e. the control surfaces generated by both neuro-fuzzy controllers are almost same), and
- Sugeno neuro-fuzzy controller in SANFIS can considerably reduce the computing time needed to infer the crisp output compared to that of the Mamdani neuro-fuzzy controller when the input values x and y are given.

3.4 TD Learning

Temporal Difference (TD) learning is popular because when using scalar rewards it can use existing supervised learning methods to tune the function approximation. When a sequence of states $< S_0, S_1, \cdots, S_f >$ of the board is given, a network visits those states sequentially until the final state S_f is encountered. Here S_0 indicates the state of the initial empty board. At the final state, a utility value (e.g., reward or punishment) is received as a final utility z.

Using the delta rule the general equation of TD(λ) is given by:

$$\Delta w_t = \alpha(P_{t+1} - P_t) \sum_{k=0}^{t} \left[\lambda^{t-k} \cdot \frac{\partial P_k}{\partial w} \right] \qquad (15)$$

where α is the learning rate and P_t is the predicted utility generated by the network at time t. We tried four values for λ: λ = 0.0, 0.3, 0.7 and 1.0.

To test the performance of TD(λ), we let two TD(λ)s play against each other (i.e. 16 games were played). The number of moves in each game was limited to 50 and a professional 9-dan (the highest rank) player assessed each games's potential result. We found that the network with higher λ (and white player) mostly wins against a lower λ network [10] when λ is not equal to 1.0. Furthermore, we found that better performance is achieved when λ = 0.0 rather than λ = 1.0 [11].

4 Results and Conclusions

For proceeding through an opening game efficiently and effectively, human players generally: (1) select a few candidate next moves (usually less than 4) from similar features they have remembered, and then (2) evaluate the efficiency of each move with deep searching for finding the best sequence of next moves with $a\ priori$ knowledge of Go concepts such as influence, group's safety, connectedness between groups etc.

Influence is a basic criteria for human players' strategic decisions in the opening game, and is strongly relating with the concepts of $thickness$ in Go theory. We used two concepts, influence and potential territory, as fuzzy inputs into the neuro-fuzzy controller. With them the maximised crisp output was induced from the Sugeno neuro-fuzzy controller in SANFIS to give the best next moves in games competing against the TD learning method.

For selecting suitable candidate next moves, we constructed an additional neural network with supervised learning. The performance of the neural network generating the set of candidate moves was promising.

For building up a neuro-fuzzy controller, we implemented the Sugeno neuro-fuzzy controller to obtain the consequent parameters. Then we constructed the Mamdani neuro-fuzzy controller to generate the target values for training the Sugeno neuro-fuzzy controller. With the target values, the Sugeno neuro-fuzzy controller was adapted

and then generated the fixed consequent parameters. Finally, we used the adapted Sugeno neuro-fuzzy controller in SANFIS with the adapted consequent parameters.

We tested the performance of the fuzzy reasoning system by playing against TD(λ)s (λ = 0.0, 0.3, 0.7 and 1.0). Eight games were played. As the evaluation of the final board positions in the opening game played was so difficult, we let a professional 9-dan player assess the games' likely results as: very favourable to black (VB), favourable to black (FB), very favourable to white (VW), favourable to white (FW), and a result which is difficult to be determined by a professional player (ND). Table 2 shows the games' potential results and Figure 6 shows a game played fuzzy reasoning (black) and TD(0.7) learning (white).

	Game 1	Game 2	Game 3	Game 4
Black	Fuzzy	TD(0.0)	Fuzzy	TD(0.3)
White	TD(0.0)	Fuzzy	TD(0.3)	Fuzzy
Result	VW	FW	VW	ND
	Game 5	Game 6	Game 7	Game 8
Black	Fuzzy	TD(0.7)	Fuzzy	TD(1.0)
White	TD(0.7)	Fuzzy	TD(1.0)	Fuzzy
Result	VB	VW	FW	VW

Table 2. Results played between the fuzzy reasoning and the TD learning.

The results reveal that the simple fuzzy reasoning system performs better than the TD(λ)s and it shows great potential to be applied to the real game of Go. This is because the neuro-fuzzy controller reflects Go knowledge with the filtered candidate moves (not the randomly generated moves) which are generated by a supervised neural network and most are good moves for an opening game.

However, the main drawback of the fuzzy reasoning system in this experiment is the lack of a richer Go knowledge. To create a more robust fuzzy reasoning system, it is recommended to analyse human reasoning: i.e. how human players acquire, organise and use Go knowledge [1].

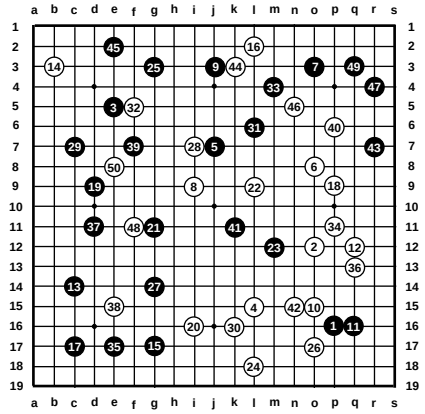


Figure 6. A game record (1-50), which is very favourable to black, played between fuzzy reasoning (black) and TD(0.7) learning (white).

References

[1] J. M. Burmeister, *Studies in Human and Computer Go: Assessing the Game of Go as a Research Domain for Cognitive Sciencer points*, PhD thesis, University of Queensland, Australia, 2000.

[2] T. Kojima, *Automatic Acquisition of Go knowledge from Game Records: Deductive and Evolutionary Approaches*, PhD thesis, University of Tokyo, Japan, 1998.

[3] M. Sugeno and G. T. Kang, Structure identification of fuzzy model, *Fuzzy Sets and Systems*, 1988, 25-33.

[4] P. K. Saha and F. W. Wehrli and B. R. Gomberg, Fuzzy Distance Transform: Theory, Algorithm, and Applications, *Computer Vision and Image Understanding*, pages 171-190, 2002.

[5] D. Hu, *Baduk Exercise Book - Standard Opening Games I and II; Two star points, Three star points, Chinese and Shusaku styles*, (Seoul: The Korean Baduk Association, 2001).

[6] R. F$\ddot{u}$ller, On Fuzzy Reasoning Scheme, *The State of the Art of Information Systems*, No. 16, 1999, 85-112.

[7] J. Binfet and B. M. Wilamowski, Microprocessor Implementation of Fuzzy Systems and Neural Networks, *IEEE*, 2001, 234-239.

[8] D. Nauck and R. Kruse, A Neuro-Fuzzy Approach to Obtain Interpretable Fuzzy Systems for Function Approximation, *IEEE*, 1998, 1106-1111.

[9] A. N$\ddot{u}$rnberger and D. Nauck and R. Kruse, Neuro-fuzzy control based on the NEFCON-model: recent developments, *In Soft Computing*, Vol 2, Springer-Verlag, 1999, 168-182.

[10] H. W. Chan and I. King and J. C. S. Lui, Performance Analysis of a New Updating Rule for TD(λ)Learning in Feedforward Networks for Position Evaluation in Go Game, *IEEE International Conference on Neural Networks*, 1996, 1716-1720.

[11] B. Lee and H. W. Guesgen and J. Baltes, The Application of TD Learning to the Opening Games of 19$\times$19 Go, *5th ICAPR Conf. on Advances in Pattern Recognition*, Calcutta, India, 2003.