An Ontology-based Fuzzy Inference System for Computer Go Applications

Chang-Shing Lee, Mei-Hui Wang, Shi-Jim Yen, Yu-Jen Chen, Cheng-Wei Chou, Guillaume Chaslot, Jean-Baptiste Hoock, Arpad Rimmel, and Hassen Doghmen

Abstract

In order to stimulate the development and research in computer Go, several Taiwanese Go players were invited to play against some famous computer Go programs. Those competitions revealed that the ontology model for Go game might resolve problems happened in the competitions. Therefore, this paper presents a Go game record ontology and Go board ontology schemes. An ontology-based fuzzy inference system is also developed to provide the regional alarm level for a Go beginner or a computer Go program in order to place the stone at the much more appropriate position. Experimental results indicate that the proposed approach is feasible for computer Go application. Hopefully, advances in the intelligent agent and the ontology model can provide a significant amount of knowledge to make a progress in computer Go program and achieve as much as computer *chess* or *Chinese chess* in the future.

Keywords: Computer Go, Fuzzy Inference, Knowledge Management, Ontology.

1. Introduction

As Go remains a challenge for computer science research [1], Monte Carlo methods are highly promising for such applications, especially for small versions of the game such as 9×9 games. Werf *et al.* [2] devised a search-based approach for playing Go on small boards. Bouzy and Cazenave [3] presented an AI-oriented survey of computer Go. Martin Müller won despite 29 handicap stones against the computer Go program *Many Faces of Go* [4]. The computer Go program *MoGo* achieved unprecedented impressive results in 19×19 game by winning with a handicap of six and seven stones against a 2P and a 9P Go player, respectively, in *Taiwan Open 2009* [15]. The computer Go program *Fuego* won a 9×9 game as White against the 9P Go player in August 2009 [18]. The latest world record involved the computer Go program *MoGoTW* winning a 9×9 game as Black against the 9P Go player in October 2009 (http://mogotw.nutn.edu.tw/), which is extremely more difficult for the computer Go program to win a top professional Go player as Black than as White.

Knowledge refers to relevant and actionable information that is based on an individual's experience [5]. Although all knowledge workers share certain characteristic activities, annotated data is obtained within a framework or ontology [6]. As a highly effective means of sharing knowledge and representing information and its semantics [7], ontology is a conceptualization of a real world domain in a human understandable, machine-readable format that consists of entities, attributes, relationships, and axioms [8]. Moreover, ontology mediation allows us to combine knowledge from the ontologies [6]. For instance, in addition to proposing a fuzzy ontology scheme for summarizing news [9], Lee *et al.* also developed an ontology-based intelligent decision support agent for project monitoring and control (PMC) process area of the capability maturity model integration (CMMI) [10]. Reformat and Ly [7] devised an ontology-based approach to provide a rich environment for expressing different information types, including perceptions.

Monte Carlo Tree Search (MCTS)-based computer Go is an important milestone for computer Go development. Minimax and alpha-beta searches are the conventional approaches adopted in computer games. However, in Go, even after pruning by patterns or rules, these approaches are clearly outperformed by MCTS [15]. Brugmann developed an original evaluation function based on Monte Carlo exploration [11]. The All-Moves-As-First (AMAF) value of a move enhances the Monte Carlo evaluation by using statistics on permutations of games. In the MCTS setting, AMAF values are usually called

Corresponding Author: Chang-Shing Lee is with the Department of Computer Science and Information Engineering, National University of Tainan, 33, Sec. 2, Shu-Lin St., Tainan, Taiwan, 70005.
E-mail: leecs@mail.nutn.edu.tw
Mei-Hui Wang and Yu-Jen Chen are with the Department of Computer Science and Information Engineering, National University of Tainan, Taiwan.
Shi-Jim Yen and Cheng-Wei Chou are with the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan.
Guillaume Chaslot is with Department of Computer Science, Maastricht University, The Netherlands.
Jean-Baptiste Hoock, Arpad Rimmel, and Hassen Doghmen are with the TAO, Lri, Univ. Paris-Sud, Inria Saclay-IDF, UMR Cnrs 8623, Bât 490, Université Paris-Sud F91405 Orsay, France.
Manuscript received 15 Nov. 2009; revised 11 Feb. 2010; accepted 8 Mar. 2010.

rapid action value estimation (RAVE) values [12]. The Monte Carlo evaluation can be used as an evaluation function in an alpha-beta engine. However, a recent and considerable improvement is the incremental construction of a tree on top of the Monte Carlo evaluation function [14]. Although still performed from the initial node, random simulations are adaptively modified in the part which is in the tree. Outside the tree, the simulation adopts the default random policy; whereas in the tree, simulations choose the move that maximizes a score combining two criteria: (1) Exploration criterion, i.e. moves seldom simulated should be more intensively simulated; and (2) Exploitation criterion, i.e. moves leading to high probabilities of winning should be simulated more often. In this incremental construction, the tree is expanded in the direction of interesting moves as estimated by the Monte-Carlo evaluation. A well-known implementation of this concept is Upper-Confidence Tree (UCT) [13, 14].

The game of Go is one of the remaining board games where the most skilled human players can still easily win against computers in 19 × 19 games. Researchers, however, have devised new performing algorithms and computers are also catching up rapidly [14]. According to game results of the *Taiwan Open 2009* [15], the ability of computers to continuously improve at this rate would result in humans losing this advantage over machines in less than a decade. The game results further demonstrate that the ability of a computer to acquire additional knowledge and strategies from professional Go players based on the constructed ontology would enable the computer Go to approach the level of a professional Go player quite rapidly. On the other hand, fuzzy logic has been widely applied to many research topics, for example, Yeh *et al.* [17] applied fuzzy logic method to analyze electrocardiogram signals for determining the heartbeat case. Based on the theories of the fuzzy control, Park *et al.* [18] proposed a balance control algorithm to acquire the pose of a biped robot from image information instead of different sensors and keep the biped robot in a stable balanced condition.

This paper presents an ontology-based fuzzy inference system for computer Go applications. The main distinction between this paper and our previous works [14-16] is that this paper proposes an ontology-based fuzzy inference system that is able to provide the regional alarm level for a Go beginner or a computer Go program as a reference when placing the stone on the Go board. In [14], it focused on the computational intelligence of *MoGo* that was revealed based on Taiwan's computer Go tournaments. The competition report on the "Human vs. Computer Go Competition," held at the 2009 IEEE International Conference on Fuzzy Systems (*FUZZ-IEEE 2009*) was introduced in [16]. In [15], a novel ontology for computer Go knowledge representation was proposed to apply to the computer Go management, and this paper is an extension of [15]. From the experimental results, it is known that with the proposed computer Go knowledge management scheme, the computer Go is expected to become much more intelligent in the near future. Experimental results also indicate that the proposed approach can increase the winning rate if a Go beginner or a computer Go program refers to the guidelines to play the game.

The rest of this paper is organized as follows. Section 2 briefly describes the definition of computer Go ontology for computer Go applications. Section 3 then introduces an ontology-based fuzzy decision support agent (*OFDSA*). Next, Section 4 introduces case studies to describe the computer Go knowledge management. Section 5 summarizes the experimental results. Conclusions are finally drawn in Section 6, along with discussions for future research.

2. Definition of Computer Go Ontology

This section introduces a novel ontology model for computer Go applications, including a domain ontology model, game record ontology, and Go board ontology.

A. *Domain Ontology Model*

Figure 1 shows the structure of the domain ontology, including a *domain layer*, *category layer*, and *concept layer* [9, 10]. The *domain layer* represents the domain name of an ontology, which consists of various categories defined by domain experts. The *category layer* defines several categories, labeled as "category *1*, category *2*, …, and category *k*." Each concept in the *concept layer* contains a concept name C_i, an attribute set $\{A_{C_i1},\cdots,A_{C_iq_i}\}$, and an operation set $\{O_{C_i1},\cdots,O_{C_iq_i}\}$ for an application domain.

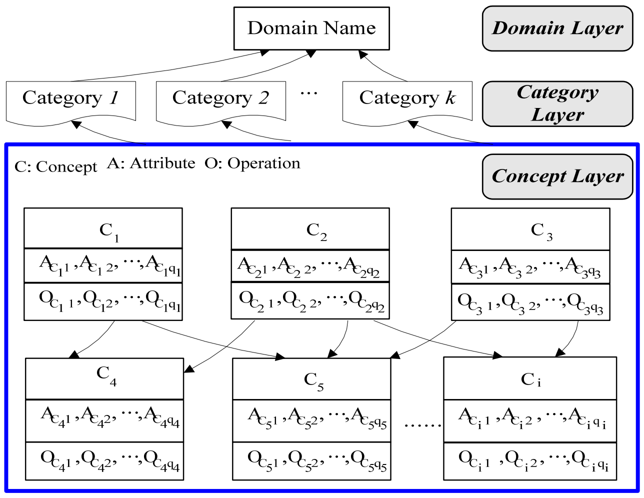
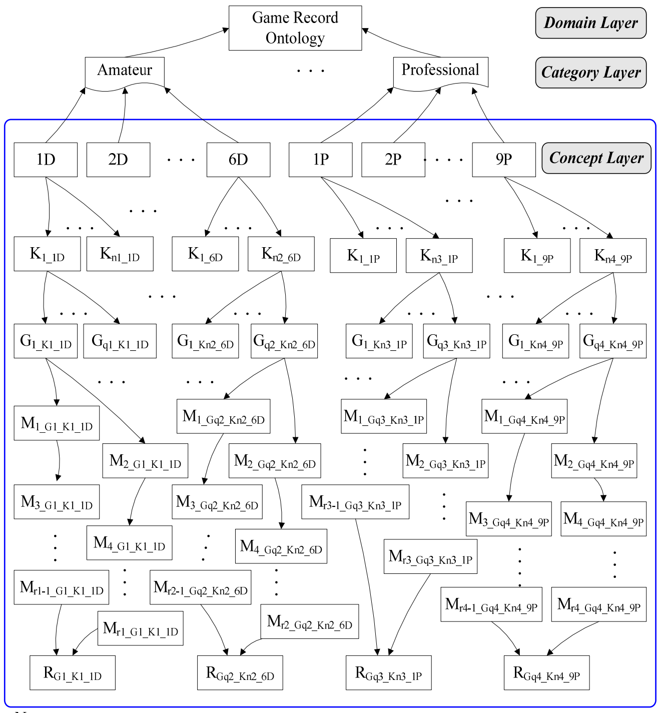

Domain Name
Domain Layer
Category 1
Category 2
…
Category k
Category Layer
C: Concept A: Attribute O: Operation
Concept Layer
C_1
$A_{C_11}, A_{C_12}, \cdots, A_{C_1q_1}$
$O_{C_11}, O_{C_12}, \cdots, O_{C_1q_1}$
C_2
$A_{C_21}, A_{C_22}, \cdots, A_{C_2q_2}$
$O_{C_21}, O_{C_22}, \cdots, O_{C_2q_2}$
C_3
$A_{C_31}, A_{C_32}, \cdots, A_{C_3q_3}$
$O_{C_31}, O_{C_32}, \cdots, O_{C_3q_3}$
C_4
$A_{C_41}, A_{C_42}, \cdots, A_{C_4q_4}$
$O_{C_41}, O_{C_42}, \cdots, O_{C_4q_4}$
C_5
$A_{C_51}, A_{C_52}, \cdots, A_{C_5q_5}$
$O_{C_51}, O_{C_52}, \cdots, O_{C_5q_5}$
……
C_i
$A_{C_i1}, A_{C_i2}, \cdots, A_{C_iq_i}$
$O_{C_i1}, O_{C_i2}, \cdots, O_{C_iq_i}$

Figure 1. Structure of the domain ontology.

B. *Game Record Ontology*

The structure of the domain ontology is applied to define the game record ontology model for computer Go knowledge management, as shown in Fig. 2.



Note:
(1) D: Dan (2) P: Professional (3) K: Kishi (Go player) (4) G: Game (5) R: Result
(6) n1, n2, n3, and n4: Total number of Go players with 1D, 2D, 1P, and 9P, respectively.
(7) q1, q2, q3, and q4: Total number of recorded games that Go players (1_1D), (n2_6D), (n3_1P), and (n4_9P) had played, respectively.
(8) r1: Total number of moves for the Game_1 that Go player (1_1D) had played.
(9) r2: Total number of moves for the Game_q2 that Go player (n2_6D) had played.
(10) r3: Total number of moves for the Game_q3 that Go player (n3_1P) had played.
(11) r4: Total number of moves for the Game_q4 that Go player (n4_9P) had played.

Figure 2. Game record ontology.

The domain name of this ontology is "*Game Record Ontology*." The categories in the *category layer* include "*Amateur*" and "*Professional*." Several concepts are located in the *concept layer*. For instance, concepts "*1D*," "*2D*," and "*6D*" represent the level of dan on the scale for an amateur Go player: 1 dan, 2 dan, and 6 dan, respectively. However, concepts "*1P*," "*2P*," and "*9P*" represent the level of dan on the scale for a professional Go player: 1 dan, 2 dan, and 9 dan, respectively. The built game record ontology stores all moves of each game played by the Go players against the computer Go programs. For instance, assume that there are *n1* Go players with 1D (K_{1_1D},…, and K_{n1_1D}). Each Go player with 1D may play total *n1* games stored in the game record ontology. Therefore, according to Fig. 2, each game of each Go player can be modeled by the built game record ontology. For instance, the games that the first Go player with 1D (K_{1_1D}) once played are represented as $G_{1\_K1\_1D}$, …, and $G_{q1\_K1\_1D}$. For the game $G_{1\_K1\_1D}$, the records of game are represented as moves *1*, *2*, …, and *r1*, that is $M_{1\_G1\_K1\_1D}$, …, and $M_{r1\_G1\_K1\_1D}$; its result is denoted as $R_{G1\_K1\_1D}$ [15].

C. *Go Board Ontology*

According to Go rules, the player with the most territory wins. However, in addition to the number of black and white stones on the game board, the position that stones were placed on the board is also a key point to influence the territory. Therefore, this paper presents a novel approach to provide the regional alarm level for a Go beginner or a computer Go program as a reference when he/she is playing the game with his/her opponent. Hopefully, with the assistance of the proposed method, a Go beginner or a computer Go program might lose the game by much fewer points than not referring to the provided regional alarm level. Based on the above discussions, the Go board is partitioned into several groups and each group is composed of nine adjacent intersections. Moreover, two adjacent groups are shared with a common edge with three adjacent intersections to strengthen how two groups are related. Such a Go board partition helps the computer to learn the matched patterns in the game of Go, enabling it to easily stimulate the playing strategies of humans.

The size of the Go board has different sizes, such as 9×9, 13×13, …, and 19×19. For instance, the 9×9 game board has 9 horizontal lines and 9 vertical lines, labeled as "1, 2, 3, 4, 5, 6, 7, 8, and 9" and "A, B, C, D, E, F, G, H, and J," respectively. Therefore, the 9×9 game board has 81 intersections. For instance, a situation in which a human plays at "A1" implies that a stone is placed at the intersection of the first horizontal line and the first vertical line. Although the Go player can place the stone on an empty intersection, the different stone-placed positions influence the game outcome differently. According to the domain ontology in Fig. 1 and the Go board partition, a Go board ontology is constructed, as shown in Fig. 3. The *domain layer* represents the domain name "*Go Board Ontology*." The *category layer* is designed based on the general Go board size, including "*9×9*", "*13×13*,"…, and "*19×19*." The *concept layer* contains concepts whose names are represented by the stone-placed position on the Go board. For instance, concept "*D6*" represents a concept located at the intersection of the vertical line "D" and the horizontal line "6" on the game board. The nine adjacent concepts are combined to form a group. Each group has a regional alarm level for a Go beginner or a computer Go program as a game-playing reference. However, the regional alarm level of one group is acquired by referring to the surrounding eight groups. That is, the surrounding eight groups influence the regional alarm level of the center group. The needed information that implements the fuzzy inference is stored into the constructed game board ontology, including fuzzy variables *Black No* (*BN*), *White No* (*WN*), *Surrounding Coefficient* (*SC*), and *Regional Alarm Level* (*RAL*). After implementing the fuzzy inference, a regional alarm level of one group is obtained.

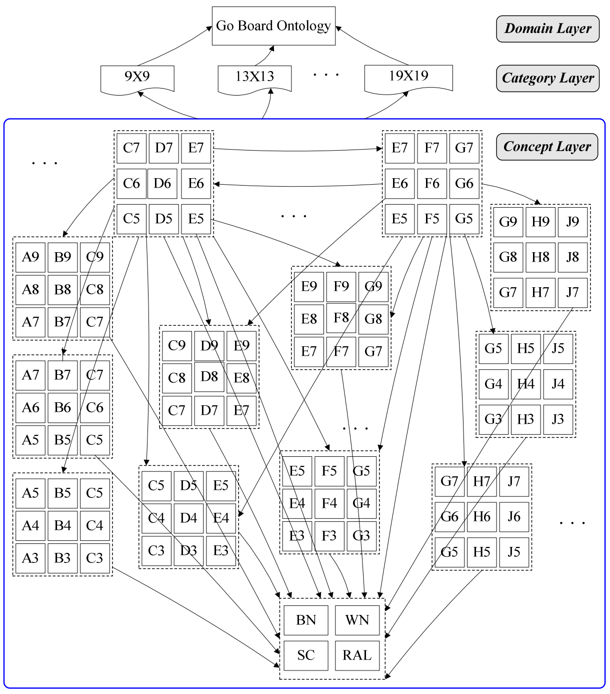


Figure 3. Go board ontology.

3. Ontology-based Fuzzy Decision Support Agent

This section introduces an ontology-based fuzzy decision support agent. The structure of the ontology-based fuzzy decision support agent is described and then the further details about the fuzzy decision support agent are given.

A. *Structure of Ontology-based Fuzzy Decision Support Agent*

Figure 4 shows the structure of the ontology-based fuzzy decision support agent (*OFDSA*), including a SGF (Smart Game Format) filtering agent, fuzzy decision support agent, Go game record repository, and ontology repository. The fuzzy decision support agent consists of three mechanisms, i.e. territory partition, fuzzy inference, and alarm presentation. The *OFDSA* functions as follows: (1) Domain experts define the Go game record ontology and the Go board ontology; in addition, they are stored into the ontology repository; (2) One Go game (Go player vs. Go player, Go player vs. computer Go program, or computer Go program vs. computer Go program), starts and a game record is gradually generated move by move; (3) Based on the constructed ontology and the generated game record, the fuzzy decision support agent infers the real-time regional alarm level of the game for supporting a Go beginner or a computer Go program to decide where to place a stone on an appropriate position during the game; (4) The inferred regional alarm levels are also stored into the Go game record repository; (5) The SGF filtering agent eliminates unnecessary messages from the generated game record; in addition, the filtered SGF files are stored into the Go game record repository; and (6) The domain experts evaluate the results of the proposed approach.

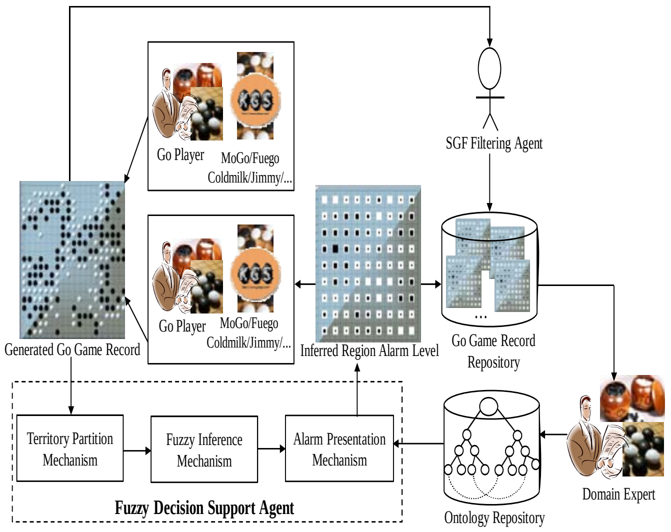

Go Player
MoGo/Fuego Coldmilk/Jimmy/...
SGF Filtering Agent
Go Player
MoGo/Fuego Coldmilk/Jimmy/...
Generated Go Game Record
Inferred Region Alarm Level
Go Game Record Repository
Territory Partition Mechanism
Fuzzy Inference Mechanism
Alarm Presentation Mechanism
Fuzzy Decision Support Agent
Ontology Repository
Domain Expert

Figure 4. Ontology-based fuzzy decision support agent structure.

B. *Territory Partition Mechanism*

In order to help a computer recognize important white-and-black patterns, the Go board is partitioned into several groups to enable a computer to more easily stimulate the playing strategy of humans. For instance, according to Fig. 5, each dashed-line frame has nine groups, with each group having nine adjacent triangular-marked intersections. Based on the patterns organized by stones placed on the forty-nine triangular-marked intersections, the circle-marked position of Fig. 5 displays the regional alarm level of group 9.

The regional alarm level of group 9 is obtained as follows: (1) Count how many black stones and white stones are placed on the triangular-marked intersections of group 1; (2) Implement the fuzzy inference mechanism to determine whether such a stone-formed pattern in group 1 makes a Go beginner or a computer Go program unfavorable to win; (3) Acquire the regional alarm level of group 1 based on the counted black stones, counted white stones, and surrounding coefficient within group 1; (4) Similarly, implement the fuzzy inference mechanism to obtain the regional alarm levels of groups 2 to 8 according to the counted black stones, counted white stones, and surrounding coefficient within groups 2 to 8, respectively; (5) Calculate the surrounding coefficient of group 9 based on the regional alarm levels of groups 1 to 8; (6) Implement the fuzzy inference mechanism to acquire the regional alarm level of group 9 according to the counted black stones, counted white stones, and surrounding coefficient within group 9; (7) Display the regional alarm level of group 9 for a Go beginner or a computer Go program in the circle-marked intersection of Fig. 5; (8) The next regional alarm level is achieved by moving the dashed-line frame to the next one; and (9) A Go beginner of or a computer Go program acquires the real-time regional alarm levels of the Go board to support his/her decisions on where to place the stone on the game board.

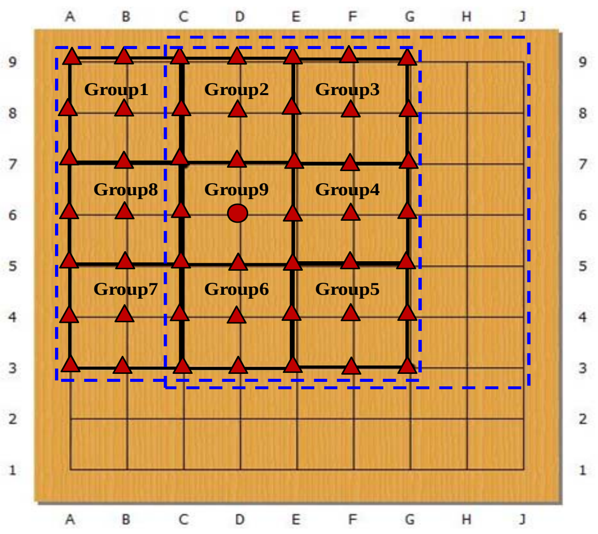


Figure 5. 9 $\times$ 9 Go board partition.

C. *Fuzzy Inference Mechanism*

This paper adopts the fuzzy inference mechanism owing to the Go game, which is characterized by high uncertainty based on the behavior of a Go player. A human playing the Go game initially observes the entire Go board and then identifies the optimal strategy from all available strategies to gain the most points or lose the fewest points within a given time limit. Additionally, exactly how many stones are on the Go board does not need to be known; instead, the entire situation on the Go board only needs to be evaluated roughly. That is, the Go player must maintain a balance between attack and defense. According to Fig. 5, the Go board is partitioned into several groups, with each group containing 9 adjacent intersections. The fuzzy inference mechanism comprises three fuzzy input variables and one fuzzy output variable. Fuzzy variables *Black No* (*BN*) and *White No* (*WN*) represent the counted black stones and white stones within one group, respectively. Both fuzzy variables *BN* and *WN* have three linguistic terms, i.e. *Low*, *Medium*, and *High*, to denote the distribution of the number of stones within one group. The third input fuzzy variable is *Surrounding Coefficient* (*SC*). For groups 2 to 8 in Fig. 5, *SC* is set to 4 for the default value. However, for group 9, *SC* denotes the dangerous level that its surrounding eight groups have caused. A higher *SC* of group 9 implies a higher risk for a Go player or a computer Go program. The output fuzzy variable is *Regional Alarm Level* (*RAL*), implying that the alarm level within the territory surrounded by one group. The regional alarm level can be very safe, safe, balanced, dangerous, and very dangerous situations for a Go player or a computer Go program.

Notably, it is recommended that a Go beginner should refer to instructions of the regional alarm level to place the stones on the intersections that are represented by the balanced state. This advice is because the balanced state typically shows where the black stones and white stones contact with each other. Additionally, to assist beginning players of Go, pre-processing information for computer Go programs can reduce the time-consuming Monte Carlo Tree Search to increase the efficiency of computer Go programs. Figure 6 displays the triangular membership function $f_A(x:a,b,c)$ of fuzzy number *A*, as used in this paper, and it is represented as *A* = [*a*, *b*, *c*]. Tables 1 and 2 list the parameters of triangular membership functions and the adopted 27 fuzzy rules, respectively. This paper adopts the MIN operator as the fuzzy conjunction operator. The MAX operation is performed to integrate the triggered rules, while the center of the area method is adopted as a defuzzification method.

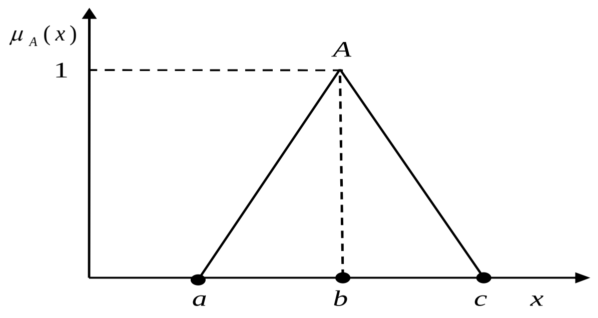


Figure 6. Triangular membership function for fuzzy set *A*.

Table 1. Parameters of triangular membership functions.

Fuzzy Variable	Linguistic Term	[*a*, *b*, *c*]
Black No (BN)	*BN_Low*	[0, 0, 2]
	BN_Medium	[1, 4.5, 8.1]
	BN_High	[5.4, 9, 9]
White No (WN)	*WN_Low*	[0, 0, 2]
	WN_Medium	[1, 4.5, 8.1]
	WN_High	[5.4, 9, 9]
Surrounding Coefficient (SC)	*SC_Low*	[0, 0, 3.2]
	SC_Medium	[0.8, 4.0, 7.2]
	SC_High	[4.8, 8, 8]
Regional Alarm Level (RAL)	*RAL_VerySafe*	[0, 0, 25]
	RAL_Safe	[0, 25, 50]
	RAL_Blanced	[25, 50, 75]
	RAL_Dangerous	[50, 75, 100]
	RAL_VeryDangerous	[75, 100, 100]

Table 2. Adopted 27 fuzzy rules.

No.	Fuzzy Variable			
	BN	*WN*	*SC*	*RAL*
1	Low	Low	Low	VerySafe
2	Low	Low	Medium	Safe
3	Low	Low	High	Balanced
4	Low	Medium	Low	Balanced
5	Low	Medium	Medium	Dangerous
6	Low	Medium	High	VeryDangerous
7	Low	High	Low	Dangerous
8	Low	High	Medium	Dangerous
9	Low	High	High	VeryDangerous
10	Medium	Low	Low	VerySafe
11	Medium	Low	Medium	Safe
12	Medium	Low	High	Balanced
13	Medium	Medium	Low	Balanced
14	Medium	Medium	Medium	Balanced
15	Medium	Medium	High	Dangerous
16	Medium	High	Low	Balanced
17	Medium	High	Medium	Dangerous
18	Medium	High	High	VeryDangerous
19	High	Low	Low	VerySafe
20	High	Low	Medium	Safe
21	High	Low	High	Balanced
22	High	Medium	Low	Safe
23	High	Medium	Medium	Balanced
24	High	Medium	High	Dangerous
25	High	High	Low	Safe
26	High	High	Medium	Balanced
27	High	High	High	VeryDangerous

D. *Alarm Presentation Mechanism*

Alarm presentation mechanism is responsible for presenting the inferred results graphically to a Go beginner or a computer Go program. With the assistance of alarm presentation, a Go beginner or a computer Go program can easily and quickly realize the overall game board situation, whether the opponent-occupied territory has a dangerous level, or whether he/she needs to give up some stones to capture safer territory. For instance, if the counted black stones are small, the counted white stones are high, and the surrounding coefficient is high, then the regional alarm level is very dangerous. This paper adopts five regional alarm levels, including very safe, safe, balanced, dangerous, and very dangerous. An icon represents each regional alarm level. The more white area this icon has, the higher risk a Go beginner or a computer Go program becomes. On the contrary, the more black area this icon has, the higher safety a Go beginner or a computer Go program becomes. Table 3 lists the appearances of these five icons. Figure 7 presents an example of the Go game record at move 27.

Table 3. Icons for the regional alarm level.

No	Icon	Semantic Meaning
1	■	Very Safe
2	◼	Safe
3	◾	Balanced
4	▪	Dangerous
5	□	Very Dangerous

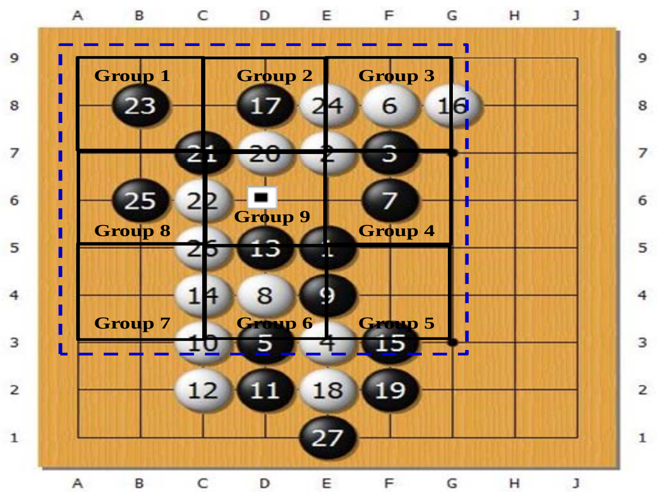


Figure 7. Game record of an example game at move 27.

Table 4. Outcomes of the fuzzy inference mechanism.

Group No	Black No	White No	Surrounding Coefficient	Region Alarm Level
1	2	0	4	0.25
2	2	3	4	0.5
3	1	4	4	0.75
4	3	1	4	0.25
5	3	0	4	0.25
6	3	4	4	0.5
7	0	3	4	0.75
8	2	2	4	0.5
9	3	4	5	0.71

Note:
(1) The default surrounding coefficient for groups 1 to 8 is 4.
(2) Based on the region alarm level value of groups 1 to 8, the surrounding coefficient value of the group 9 is acquired. If the region alarm level value of groups 1 to 8 is larger or equal 0.5, the surrounding coefficient value of the group 9 is plus 1.

Table 4 lists the outcomes of the fuzzy inference mechanism when Fig. 7 is used as an example. According to this table, the regional alarm level of group 9 in Fig. 7 is 0.71, and the semantic meaning of “0.71” is “Balanced.” Therefore, position “D6” in Fig. 7 is denoted by a balanced icon, “◾”. Figure 8 summarizes the results of the alarm representation mechanism of Fig. 7, indicating that although the bottom left corner is very safe, the right side of the Go board is dangerous to lose the game for a Go beginner or a computer Go computer.

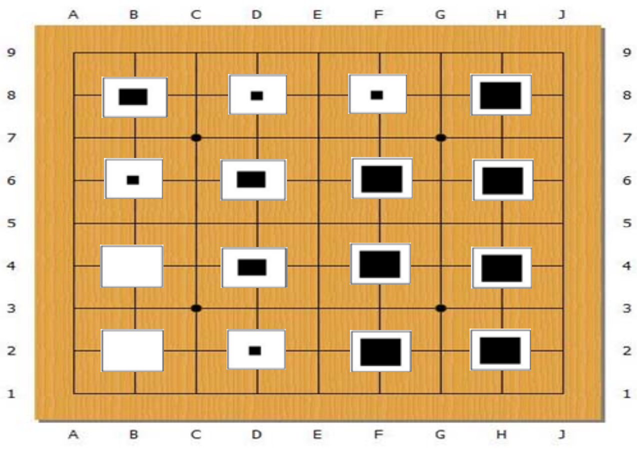


Figure 8. Results of the alarm representation mechanism.

4. Computer Go Knowledge Management

This section introduces a computer Go knowledge management. The knowledge management process is described and then recent computer Go competitions are introduced. Finally, the case studies are given.

A. *Knowledge Management Process*

Applying pattern knowledge and accumulated positional knowledge allows a Go player to exclude bad moves from consideration and recognize quickly and stop exploring hopeless positions during play. The ability of a computer Go program to utilize domain knowledge from domain experts would make the computer Go programs more efficient and reliable.

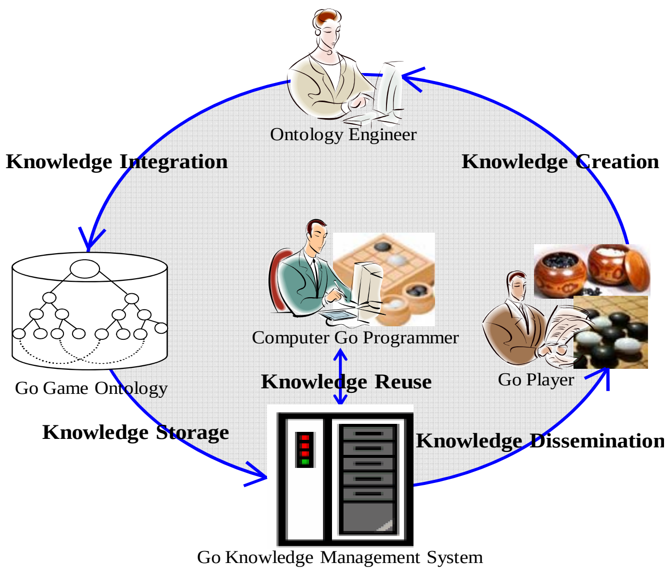


Figure 9. Knowledge management process for computer Go.

Figure 9 shows how the Go knowledge management is constructed, which contains five sub-processes, i.e. knowledge creation, knowledge integration, knowledge storage, knowledge dissemination, and knowledge reuse. Some operations shown in Fig. 9 are described as follows. First, the major knowledge creation transaction is between a Go player and an ontology engineer. This transaction can also be constructed by the computer Go program. Ontology engineering identifies and acquires Go domain knowledge from a Go player through communication with each other to create knowledge. Second, the ontology engineer integrates the acquired knowledge to construct a Go ontology. Third, the Go domain knowledge is stored into the Go knowledge management system. Fourth, the knowledge is then made available to the computer Go programmer and other Go players through knowledge reuse and knowledge dissemination. Finally, the computer Go programmer develops a novel game record ontology for computer Go knowledge management to coordinate with Go players [15].

B. *Introduction of Computer Go Competition*

Due to advances in the computational intelligence, computer Go has made considerable progress in the recent decade. Programs are competitive at the professional level in 9×9 Go. The "*2009 Invited Games for MoGo vs. Taiwan Professional Go Players* (*Taiwan Open 2009*)" was held on February 10-13, 2009. The organizers invited three professional Go players, including Chun-Hsun Chou (9P, top pro player winner of the LG Cup 2007), Shih Chin (2P), and Li-Chen Chien (1P), as well as several amateur Go players, including Cheng-Wen Dong, Shi-Jim Yen, Shang-Rong Tsai, and Biing-Shiun Luoh, to compete with *MoGo* in the competition [15]. Additionally, the "Panel, Invited Sessions, and Human vs. Computer Go Competition" was held on August 19-23, 2009 at the *FUZZ-IEEE 2009*. During this competition, some global leading computer Go programs, including *MoGo*, *Fuego*, *Zen*, and *Many Faces of Go*, were invited to play with Chun-Hsun Chou (9P) and Shen-Su Chang (6D) [16]. Moreover, Chun-Hsun Chou (9P) was also invited to play against *MoGoTW*, during a press conference in Taipei, Taiwan. By holding computer Go competitions against humans, some advances in the artificial intelligence have been made. For instance, (1) *MoGo* won a 9P Go player in 19×19 game with handicap seven stones, (2) *Fuego* won as White against a 9P Go player in 9×9 game, and (3) *MoGoTW* won as Black against a 9P Go player in 9×9 game. Table 5 lists a brief introduction of some computer Go programs.

Table 5. Brief introductions to some computer Go programs.

No	Computer Go Program Name	Brief Introduction
1	*MoGo* (France)	Gold modal of 2007 Computer Olympiad, Go (19 × 19)
2	*Fuego* (Canada)	Gold modal of 2009 Computer Olympiad, Go (9 × 9)
3	*Zen* (Japan)	Gold modal of 2009 Computer Olympiad, Go (19 × 19)
4	*Many Faces of Go* (USA)	Gold modal of 2008 Computer Olympiad, Go (9 × 9 and 19 × 19)
5	*Jimmy* (Taiwan)	Developed by Prof. Shi-Jim Yen
6	*Coldmilk* (Taiwan)	Based on Jimmy and developed by the Artificial Intelligence Laboratory (AI Lab) of National Dong Hwa University (NDHU)
7	*MoGoTW* (France-Taiwan)	Based on *MoGo* 4.86 Soissons plus the modifications developed jointly with *MoGo* team members and Taiwan team members

C. *Case Studies of Computer Go Competition*

This subsection discusses eight 9 × 9 games of *Taiwan Open 2009*. Table 6 lists the profiles of partial Go players who competed against *MoGo*. The game adopted the Chinese rule, and komi was 7.5. During the tournament, *MoGo* ran on a supercomputer Huygens. Table 7 summarizes the results of the discussed 9 × 9 games.

Table 6. Profiles of partial Go players competing with *MoGo*.

No	Name	Age	Sex	Dan grade
1	Chun-Hsun Chou	28	Male	9P
2	Shih Chin	22	Male	2P
3	Shi-Jim Yen	41	Male	6D

Table 7. Results of discussed 9 × 9 games.

No	Date	Time/side (min)	White	Black	Result
1	02/10/2009	30	***MoGo***	Yen	W+Resign
2	02/10/2009	50	Yen	***MoGo***	B+1.5
3	02/12/2009	50	***MoGo***	Chin	W+0.5
4	02/12/2009	50	**Chin**	*MoGo*	W+Resign
5	02/10/2009	50	**Chou**	*MoGo*	W+Resign
6	02/13/2009	50	**Chou**	*MoGo*	W+Resign
7	02/10/2009	50	*MoGo*	**Chou**	B+Resign
8	02/13/2009	50	*MoGo*	**Chou**	B+Resign

Figures 10 and 11 display the recorded moves of games 5 and 6, respectively. According to these figures, *MoGo* played the same sequences and positions against Chou for moves 1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29, 31, 33, and 35. Also, games 7 and 8 were nearly identical, especially from move 1 to move 19. The Go player just reproduced the same games. During game 6 against Chou, *MoGo* played the same moves as the ones during game 5 against Chou. However, from a human perspective, *MoGo* should have played different moves on game 6 to have an opportunity to defeat Chou. The need to change the strategy in order to have the opportunity to win is a relatively simple inference for a human. This problem could possibly be solved if *MoGo* can acquire further knowledge experience of playing with humans. Therefore, a game record ontology is constructed in this paper for computer Go knowledge management. Figure 12 shows the *Taiwan Open 2009* Go knowledge management ontology. First, the domain name is "*Taiwan Open 2009 Go Players*." There is one 5D amateur concept "*Cheng-Wen Dong*" and three 6D amateur concepts, including "*Shi-Jim Yen*," "*Shang-Rong Tsai*," and "*Biing-Shiun Luoh*." Concepts "*Li-Chen Chien*," "*Shih Chin*," and "*Chun-Hsun Chou*," which are 1P, 2P, and 9P professional Go players, respectively [15].

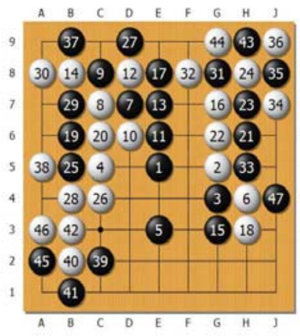


Figure 10. Game 5.

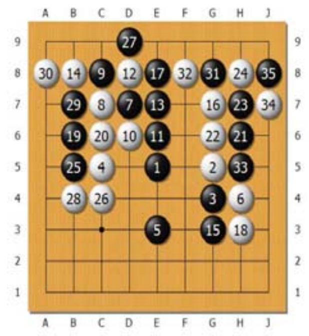


Figure 11. Game 6.

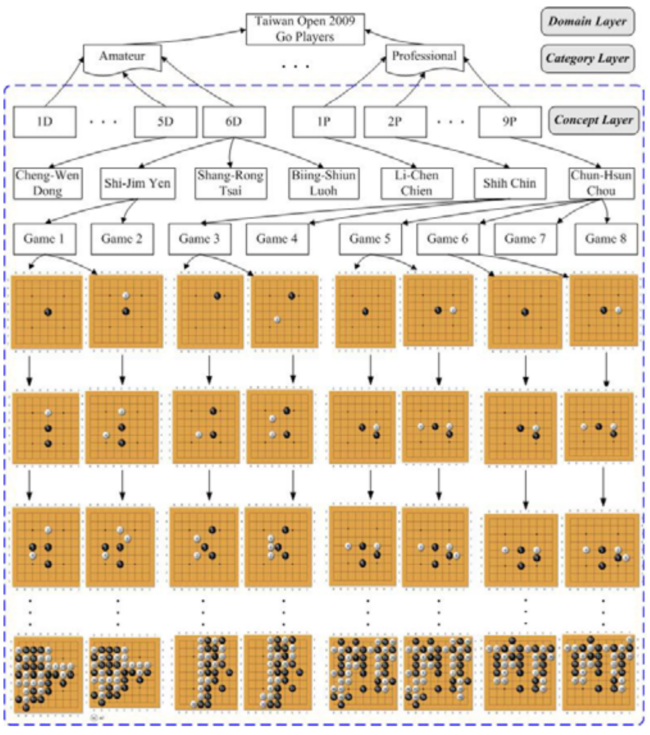


Figure 12. *Taiwan Open 2009* Go knowledge management ontology.

5. Experimental Results

This paper has constructed an experimental platform at National University of Tainan (Taiwan), INRIA (France), and National Dong Hwa University (Taiwan) to evaluate the performance of the proposed approach. The first experiment evaluates whether if the performance of the computer Go program *Coldmilk* is enhanced when *Coldmilk* refers to the *OFDSA*-provided regional alarm level as the guidelines of searching directions. The test environment is as follows: (1) The test version of *Coldmilk* is 2.22; (2) *Fuego* is the opponent*;* (3) The number of simulations is 3000 for both sides; (4) The total number of played games is 600; and (5) *Coldmilk* is Black for 300 games, and *Fuego* is Black for another 300 games. In this experiment, various membership functions along with different default *SC* values (4, 6, or 8) are selected to evaluate the *OFDSA* performance.

In Table 8, the membership function (MF) 1 is the standard one used in *OFDSA*, and the marked-bold parameters for MFs 2 to 7 denote the differences from the ones of the MF 1. Table 9 displays the variability in *Coldmilk* winning rate vs. *Fuego*, indicating that the best winning rate is 46.2% and the error range is $\pm 2\%$ when parameters of MF 6 are used in the fuzzy inference mechanism. However, if the standard parameters (MF 1) are utilized, the winning rate ranges from 39.2% to 43.1%. Consequently, Table 9 reveals that adjusting the shapes of the membership functions affects the results of the winning rate. We believe that embedding the machine learning methods such as the genetic algorithm or the neural network to the *OFDSA* might enhance the performance of the winning rate of *Coldmilk* in the future.

In the second experiment, a 5D Go player (Cheng-Wen Dong, 70-year-old) as White was invited to compete against 9K Go player (Yu-Jen Chen, 23-year-old) as Black in 19×19 game with four handicap stones. Figure 13(a) shows the game record, while Fig. 13(b) shows the *OFDSA*-provided alarm representation. Figure 13 reveals the following: (1) Fig. 13 (a) indicates that the area surrounded by the dashed-line frame is Black's territory. White cannot be alive within this territory. According to Fig. 13(b), *OFDSA* is feasible for stimulating similar human-playing thinking because the dashed-line frame is surrounded by the "Safe" icons; (2) Figure 13(b) indicates that the territory marked by "Balanced" icons is exactly what the contacts between Black and White are. Hence, the adjacent intersections around the "Balanced" icon could be an indicator of Black placing the stone on the game board during the game; (3) A situation in which the territory is marked "VeryDangerous" icons implies that this situation is very unfavorable for Black. According to Fig. 13 (a), the top-left corner and bottom-right corners are White territory. Figure 13 (b) also reveals that the "VeryDangerous" icons are represented in the top-left corner and bottom-right corner of the game board. Based on the above discussions, we can infer that the provided regional alarm levels coincide with the human's approach to evaluate the current situation of the game board.

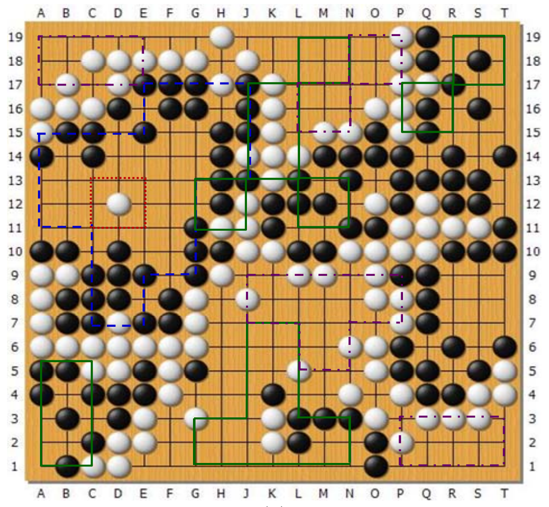
Table 8. Various different membership functions.

Fuzzy Variable	*BN*	*WN*	*SC*	*RAL*
Linguistic Term / MF No	*Low* *Medium* *High*	*Low* *Medium* *High*	*Low* *Medium* *High*	*VerySafe* *Safe* *Balanced* *Dangerous* *VeryDangerous*
1	[0, 0, 2] [1, 4.5, 8.1] [5.4, 9, 9]	[0, 0, 2] [1, 4.5, 8.1] [5.4, 9, 9]	[0, 0, 3.2] [0.8, 4.0, 7.2] [4.8, 8, 8]	[0, 0, 25] [0, 25, 50] [25, 50, 75] [50, 75, 100] [75, 100, 100]
2	**[0, 0, 8]** **[5, 7, 9]** **[8, 9, 9]**	**[0, 0, 1]** **[0, 2, 4]** **[1, 9, 9]**	[0, 0, 3.2] [0.8, 4.0, 7.2] [4.8, 8, 8]	[0, 0, 25] [0, 25, 50] **[2.5, 50, 75]** [50, 75, 100] [75, 100, 100]
3	[0, 0, 2] [1, 4.5, 8.1] [5.4, 9, 9]	[0, 0, 2] [1, 4.5, 8.1] [5.4, 9, 9]	**[0, 0, 1]** **[0, 1, 2]** **[1, 8, 8]**	[0, 0, 25] [0, 25, 50] **[2.5, 50, 75]** [50, 75, 100] [75, 100, 100]
4	**[0, 0, 8]** **[5, 7, 9]** **[8, 9, 9]**	**[0, 0, 1]** **[0, 2, 4]** **[1, 9, 9]**	**[0, 0, 1]** **[0, 1, 2]** **[1, 8, 8]**	[0, 0, 25] [0, 25, 50] **[2.5, 50, 75]** [50, 75, 100] [75, 100, 100]
5	**[0, 0, 1]** **[0, 1, 2]** **[1, 9, 9]**	**[0, 0, 1]** **[0, 2, 4]** **[1, 9, 9]**	[0, 0, 3.2] [0.8, 4.0, 7.2] [4.8, 8, 8]	[0, 0, 25] [0, 25, 50] [25, 50, 75] [50, 75, 100] [75, 100, 100]
6	[0, 0, 2] [1, 4.5, 8.1] [5.4, 9, 9]	[0, 0, 2] [1, 4.5, 8.1] [5.4, 9, 9]	[0, 0, 3.2] [0.8, 4.0, 7.2] [4.8, 8, 8]	[0, 0, 25] [0, 25, 50] **[2.5, 50, 75]** [50, 75, 100] [75, 100, 100]
7	[0, 0, 2] [1, 4.5, 8.1] [5.4, 9, 9]	[0, 0, 2] [1, 4.5, 8.1] [5.4, 9, 9]	[0, 0, 3.2] [0.8, 4.0, 7.2] [4.8, 8, 8]	[0, 0, **2.5**] [0, 25, 50] [**2.5**, 50, 75] [50, 75, 100] [75, 100, 100]

Table 9. Variability in the *Coldmilk* winning rate vs. *Fuego*.

OFDSA	Winning Rate (±2%)		
No	42.8		
	MF No	Default *SC*	Winning Rate (±2%)
Yes	1	4	39.2
	1	6	42.0
	1	8	43.1
	2	8	40.7
	3	8	41.0
	4	8	43.1
	5	4	42.5
	6	**8**	**46.2**
	6	8	43.8
	7	8	40.4

The final experiment involves observing whether the *OFDSA* can successfully guide Go beginners to make them lose the game by significantly fewer points when their opponent's is stronger than them. Four Go players play with *MoGo* (MoGo_release3) running on a Windows PC with 4 cores. Meanwhile, another beginning Go player competes against *MoGo* (v4.86) running on a super computer with 16 cores through a KGS server. Komi is 7.5 and the Chinese rule is adopted. Table 10 lists the game results of beginning Go players vs. *MoGo*, where "GP" refers to "Go Player." This table reveals the variance in lost points under the conditions of with and without the guidelines of the *OFDSA*, where the lost points are reduced if the Go beginning players refer to the regional alarm level provided by the *OFDSA* to play the games. Figures 14(a) and 14(b) display the game record and generated regional alarm level of game 20 at move 32, respectively. Figure 14(b) indicates that the top-right corner and bottom side are very dangerous for Go player 5, which matches with the situation of the game board in Fig. 14(a).


(a)

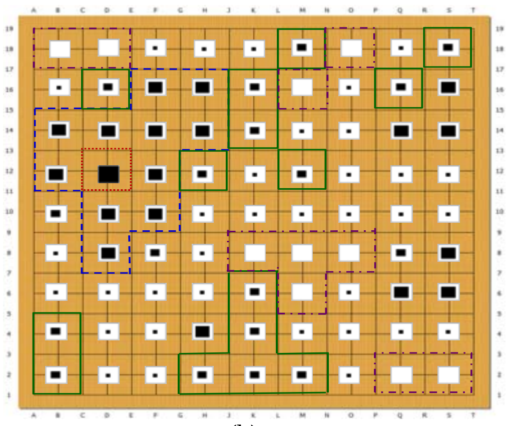

(b)

Figure 13. (a) Game record (b) Regional alarm levels.

Table 10. Game results of Go players vs. *MoGo*.

Time Per Side	Game No	White	Black	Result *OFDSA* No	Result *OFDSA* Yes
5min	1	*MoGo*	GP1	W+88.5	
	2	*MoGo*	GP1		W+10.5
	3	*MoGo*	GP1	W+88.5	
	4	*MoGo*	GP1		W+56.5
	5	GP1	*MoGo*	B+72.5	
	6	GP1	*MoGo*		B+72.5
	7	GP1	*MoGo*	B+72.5	
	8	GP1	*MoGo*		B+52.5
	9	*MoGo*	GP2	W+88.5	
	10	*MoGo*	GP2		W+17.5
	11	*MoGo*	GP2	W+39.5	
	12	*MoGo*	GP2		W+87.5
	13	*MoGo*	GP2	W+87.5	
	14	*MoGo*	GP2		W+6.5
	15	*MoGo*	GP3	W+63.5	
	16	*MoGo*	GP3		W+60.5
	17	*MoGo*	GP4	W+44.5	
	18	*MoGo*	GP4		W+20.5
10min	19	*MoGo*	GP5	W+18.5	
	20	*MoGo*	GP5		W+12.5
	21	GP5	*MoGo*	B+23.5	
	22	GP5	*MoGo*		B+3.5

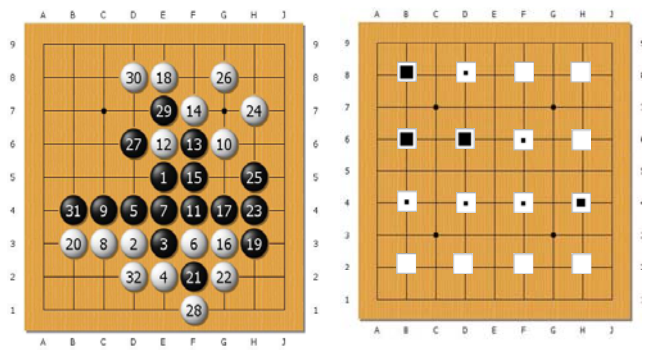

(a) (b)

Figure 14. (a) Game record and (b) regional alarm level of game 20 at move 32.

6. Conclusions

This paper presents a novel ontology-based fuzzy inference approach for computer Go applications. Experimental results indicate that the winning rate can be increased if the proper membership functions are identified. Additionally, some regional alarm levels provide a valuable reference for either a Go beginner or a computer Go program when playing the game. However, the proposed approach still has certain limitations so that future research should (1) combine the proposed method with MCTS method to reduce expensive searching cost by searching for the optimal move within the dangerous region, (2) add machine learning methods such as a genetic algorithm or neural network to increase the winning rate, (3) enhance more closely examining human-playing strategies of the constructed ontology for use as reference for giving up stones, (4) add the weights to the territory of the Go board to strengthen the relationship among groups, and (5) display the regional alarm levels in the semantic expressions of humans.

Acknowledgment

The authors would like to thank the National Science Council of Taiwan for financially supporting this research(NSC97-2221-E-024-011-MY2,NSC98-2221-E-024-009-MY3, and NCS99-2923-E-024-003-MY3). Additionally, the authors also would like to thank the Go players for their playing against *MoGo* as well as Dr. Olivier Teytaud, Mr. Chun-Hsun Chou, Prof. Shang-Rong Tsai, and Prof. Shun-Chin Hsu for their help with this paper. Ted Knoy is appreciated for his editorial assistance

References

[1] S. M. Lucas and G. Kendall, “Evolutionary computation and games,” *IEEE Computational Intelligence Magazine*, vol. 1, no. 1, pp. 10-18, 2006.

[2] E. van der Werf, J. van den Herik, and J. Uiterwijk, “Solving Go on small boards,” *International Computer Games Association (ICGA) Journal*, vol. 26, no. 2, pp. 92-107, 2003.

[3] B. Bouzy and T. Cazenave, “Computer Go: an AI-oriented survey,” *Artificial Intelligence Journal*, vol. 132, no. 1, pp. 39-103, 2001.

[4] M. Müller, “Computer Go,” *Artificial Intelligence*, vol. 134, no. 1-2, pp. 145-179, 2002.

[5] B. Ghosh and J. E. Scott, “Comparing knowledge management in health-care and technical support organizations,” *IEEE Trans. on Information Technology in Biomedicine*, vol. 9, no. 2, pp. 162-168, 2005.

[6] P. Warren, “Knowledge management and the semantic web: from scenario to technology,” *IEEE Intelligent Systems*, vol. 21, no. 1, pp. 53-59, 2006.

[7] M. Reformat and C. Ly, “Ontological approach to development of computing with words based systems,” *International Journal of Approximate Reasoning*, vol. 50, no. 1, pp. 72-91, 2009.

[8] Q. T. Tho, S. C. Hui, A. C. M. Fong, and T. H. Cao, “Automatic fuzzy ontology generation for semantic web,” *IEEE Trans. on Knowledge and Data Engineering*, vol. 18, no. 6, pp. 842-856, 2006.

[9] C. S. Lee, Z. W. Jian, and L. K. Huang, “A fuzzy ontology and its application to news summarization,” *IEEE Trans. on Systems, Man and Cybernetics Part B*, vol. 35, no. 5, pp. 859-880, 2005.

[10] C. S. Lee, M. H. Wang, and J. J. Chen, “Ontology-based intelligent decision support agent for CMMI project monitoring and control,” *International Journal of Approximate Reasoning*, vol. 48, no. 1, pp. 62-76, 2008.

[11] B. Brugmann, Monte Carlo Go, 1993. Online at http://www.ideanest.com/vegos/MonteCarloGo.pdf

[12] S. Gelly and S. Silver, “Combining online and offline knowledge in UCT,” *In Proc. of the 24th International Conference on Machine Learning* (*ICML 2007*), pp. 273-280, New York, USA, Jun. 20-24, 2007.

[13] L. Kocsis and C. Szepesvari, “Bandit-based Monte-Carlo planning,” Lecture Notes in Computer Science, vol. 4212, pp. 282-293, 2006.

[14] C. S. Lee, M. H. Wang, C. Chaslot, J. B. Hoock, A. Rimmel, O. Teytaud, S. R. Tsai, S. C. Hsu, and T. P. Hong, “The computational intelligence of MoGo revealed in Taiwan’s computer Go tournaments,” *IEEE Trans. on Computational Intelligence and AI in Games*, vol. 1, no. 1, pp. 73-89, 2009.

[15] C. S. Lee, M. H. Wang, T. P. Hong, G. Chaslot, J. B. Hoock, A. Rimmel, O. Teytaud, and Y. H. Kuo, “A novel ontology for computer Go knowledge management,” *In Proc. of the 2009 IEEE International Conference on Fuzzy System* (*FUZZ-IEEE 2009*), pp. 1056-1061, Jeju Island, Korea, Aug. 19-24, 2009.

[16] S. J. Yen, C. S. Lee, and O. Teytaud, “Human vs. computer Go competition in FUZZ-IEEE 2009,” International Computer Games Association (ICGA) Journal, vol. 32, no. 3, pp. 178-180, Sept. 2009.

[17] Y. C. Yeh, W. J. Wang, and C. W. Chiou, “Heartbeat case determination using fuzzy logic method on ECG signals,” *International Journal of Fuzzy Systems*, vol. 11, no. 4, pp. 250-261, 2009.

[18] S. Park, Y. Han, and H. Hahn, “Fuzzy controller based biped robot balance control using 3D image,” *International Journal of Fuzzy Systems*, vol. 11, no. 3, pp. 202-212, 2009.

Chang-Shing Lee (SM’09) received the Ph.D. degree in computer science and information engineering from the National Cheng Kung University, Tainan, Taiwan, in 1998. He is currently a Professor with the Department of Computer Science and Information Engineering, National University of Tainan, Taiwan, where he is the Director of the Computer Center. He is also an Associate Editor of the *Journal of Ambient Intelligence and Humanized Computing*. He is a member of the Editorial Board for *Applied Intelligence*, the *Journal of Advanced Computational Intelligence and Intelligent Informatics*, and the *Open Cybernetics and Systemics Journal*. He is a Guest Editor for the *Applied Intelligence Journal*, the *International Journal of Intelligent System*, the *International Journal of Fuzzy Systems*, and the *Journal of Internet Technology*. His current research interests include ontology applications, knowledge management, capability maturity model integration, meeting scheduling, and artificial intelligence, and he is also interested in intelligent agents, web services, fuzzy theory and applications, genetic algorithms, and image processing. He also holds several patents on ontology engineering, document classification, image filtering, and healthcare. Dr. Lee has been the Emergent Technologies Technical Committee (ETTC) Chair of the IEEE Computational Intelligence Society (CIS) since 2009. During 2008, he was the ETTC Vice Chair of the IEEE CIS. He is an Associate Editor and a Guest Editor of the IEEE TRANSACTIONS ON COMPUTATIONAL INTELLIGENCE AND AI IN GAMES. He is also a member of the Program Committees of more than 40 conferences. He is a member of the Taiwanese Association for Artificial Intelligence and the Software Engineering Association Taiwan.

Mei-Hui Wang received the B.S. degree in biomedical engineering from the Chung Yuan Christian University, Chung-Li, Taiwan, in 1993 and the M.S. degree in electrical engineering from the Yuan Ze University, Chung-Li, Taiwan, in 1995. From July 1995 to June 2005, she worked for the Delta Electronics, Inc., Chung-Li, Taiwan, as a Senior Firmware Engineer. Currently, she is a Researcher at the Ontology Application & Software Engineering (OASE) Laboratory, Department of Computer Science and Information Engineering, National University of Tainan, Tainan, Taiwan. Her research interests include intelligent agent, ontology engineering, and image processing.

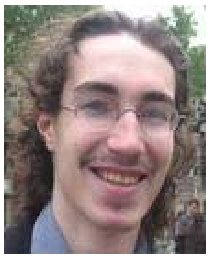
Shi-Jim Yen received his Ph.D. degree in Computer Science and Information Engineering from the National Taiwan University, Taiwan. He is currently an Associate Professor with the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan. He specializes in artificial intelligence (AI) and computer games. In these areas, he has published over 30 papers in international journals or conference proceedings. He has been an amateur Go player with the skill about 6 Dan since 2002. He served as a workshop chair on the 5th international conference on Grid and Pervasive Computing in 2010, and a workshop chair of 2010 International Taiwanese Association for Artificial Intelligence (TAAI) conference. He is the Chair of the IEEE Computational Intelligence Society (CIS) Emergent Technologies Technical Committee (ETTC) Task Force on Emerging Technologies for Computer Go in 2010.

Yu-Jen Chen was born in Kaohsiung, Taiwan in 1986. He received the B.S. degree in Department of Computer Science from the Hsuan Chuang University, Hsinchu, Taiwan, in 2009. He is currently working toward the M.S. degree in Computer Science and Information Engineering at National University of Tainan, Taiwan. His research interests include ontology engineering, artificial intelligence, and fuzzy theory.

Cheng-Wei Chou received his M.S. degree in Computer Science and Information Engineering from the Fu Jen Catholic University, Taiwan. Currently, he is working as Ph.D. student at the Department of Computer Science and Information Engineering, National Dong Hwa University, Taiwan. He specializes in computer games. His computer Go program has participated in multiple international events. He has been an amateur Go player with the skill about 4 Dan since 2008.

Guillaume Chaslot received the B.S. degree from the Ecole Centrale de Lille, France, in 2005, and the M.S. degree in Computer Science and Artificial Intelligence from the University of Lille 1, France, in 2005. Currently, he is working as a Ph.D. student at the Maastricht University, The Netherlands. His research topic focuses on the development of Monte-Carlo Tree Search techniques, and the field of Machine Learning and Artificial Intelligence.

Jean-Baptiste Hoock was born in Creil, France in 1983. He received the master in computer science and image processing from Caen University and engineering diploma from the ENSICAEN Engineer School, Caen, France, in 2006. He is currently working as an engineer at University of South Paris, France. His research interests include Monte-Carlo Tree Search with the application in the game of Go.



Arpad Rimmel was born in Paris, France in 1983. He received the M.S. degree in Computer Science from the University of South Paris in 2005 and received the Ph.D. degree in Artificial Intelligence at the University of South Paris in 2009. His research interests include reinforcement learning and Monte-Carlo tree search with application to the game of Go.

Hassen Doghmen was born in Bizerte, Tunisia in 1984. He received the M.S. degree in Computer Science and Systems in fields of modeling, optimization, programming, and services. He is currently working as an engineer at INRIA-SACLAY, France. His research interests include Monte-Carlo Tree Search with the application in the game of Go.