On Competency of Go Players: A Computational Approach

Amr Ahmed Sabry Abdel-Rahman Ghoneim

B.Sc. *(Computer Science & Information Systems)* Helwan University, Cairo - Egypt

M.Sc. *(Computer Science)* Helwan University, Cairo - Egypt

A thesis submitted in partial fulfilment of the requirements
for the degree of Doctor of Philosophy at the
School of Engineering & Information Technology
the University of New South Wales at
the Australian Defence Force Academy

March 2012

[This Page is Intentionally Left Blank]

Abstract

Complex situations are very much context dependent, thus agents – whether human or artificial – need to attain an awareness based on their present situation. An essential part of that awareness is the accurate and effective perception and understanding of the set of knowledge, skills, and characteristics that are needed to allow an agent to perform a specific task with high performance, or what we would like to name, *Competency Awareness*. The development of this awareness is essential for any further development of an agent's capabilities. This can be assisted by identifying the limitations in the so far developed expertise and consequently engaging in training processes that add the necessary knowledge to overcome those limitations. However, current approaches of competency and situation awareness rely on manual, lengthy, subjective, and intrusive techniques, rendering those approaches as extremely troublesome and ineffective when it comes to developing computerized agents within complex scenarios. Within the context of computer Go, which is currently a grand challenge to Artificial Intelligence, these issues have led to substantial bottlenecks that need to be addressed in order to achieve further improvements. Thus, the underlying principle of this work is that of the development of an automated, objective and non-intrusive methodology of Competency Assessment of decision-makers, which will practically aid in the understanding and provision of effective guidance to the development of improved decision-making capabilities, specifically within computer agents.

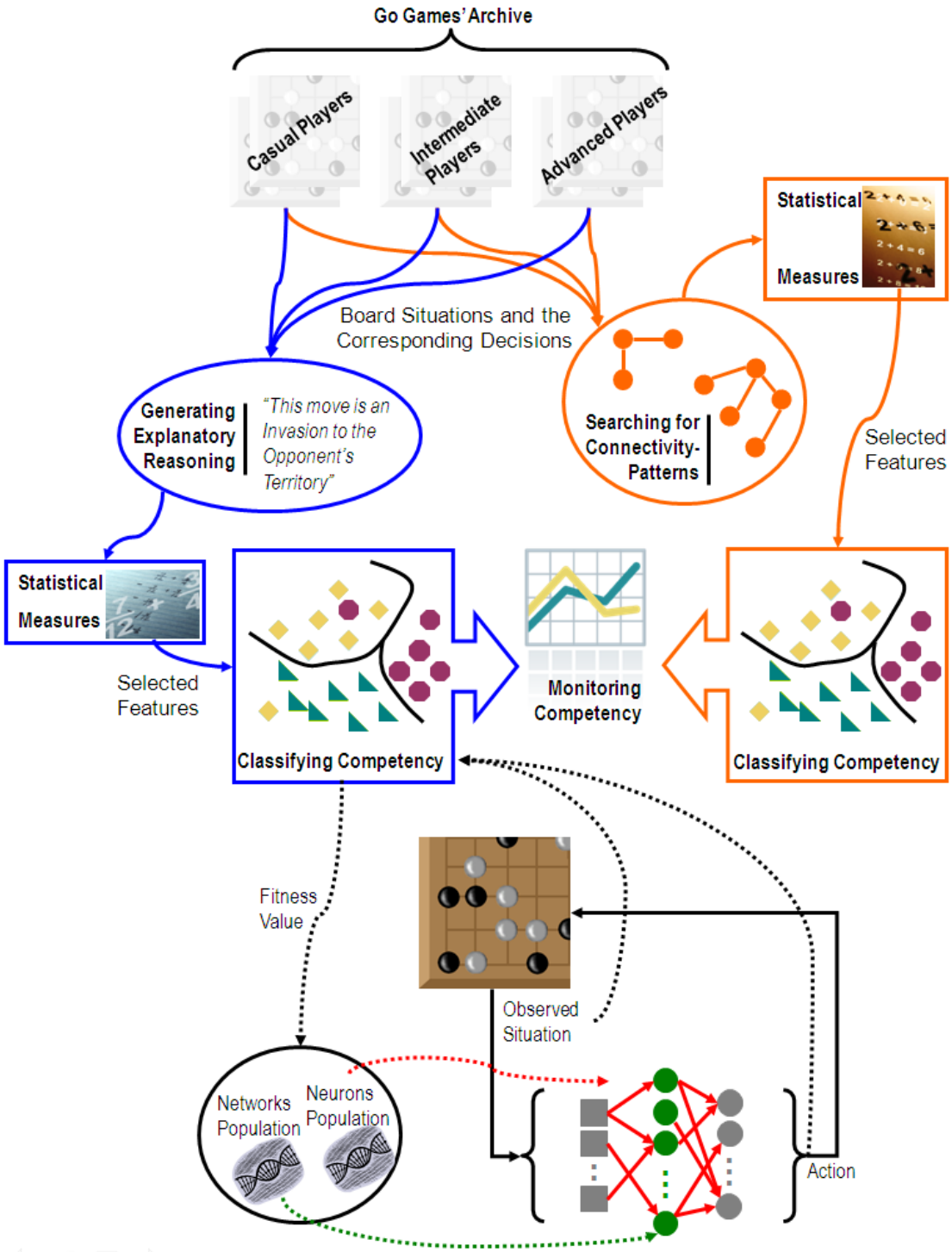
In this study, we propose a framework whereby a computational environment is used to study and assess the competency of a decision maker. We use the game of GO to demonstrate this functionality in an environment in which hundreds of human-played GO games are analysed. In order to validate the proposed framework, a series of experiments on a wide range of problems have been conducted. These experiments automatically: (1) measure and monitor the competency of Human Go players, (2) reveal and monitor the dynamics of Neuro-Evolution, and (3) integrate strategic domain knowledge into evolutionary algorithms. The experimental results show that: (1) the

proposed framework is effective in measuring and monitoring the strategic competencies of human Go players and evolved Go neuro-players, and (2) is effective in guiding the development of improved Go neuro-players when compared to traditional approaches that lacked the integration of a strategic competency measurement.

Keywords

Complex Scenarios, Decision Making, Strategic Reasoning, Competency Assessment, Situation Awareness, Situation Management, Skill Assessment, Computational Intelligence, Random Forests, Artificial Neural Networks, Evolutionary Computation, Neuro-Evolution, Rules Extraction, Network Motifs, Mind Games, Strategic Games, Board Games, Game Playing, Go, Computer Go

Graphical Abstract



[This Page is Intentionally Left Blank]

Acknowledgment

It would not have been possible to complete my PhD without the support of so many wonderful people.

First of all, I would like to express my sincere thanks to my principal supervisor, Prof Hussein Abbass, for his wisdom and understanding, exchanging ideas, and his guidance in all aspects of my work, providing me great help on various topics.

I also would like to express my sincere thanks to my co-supervisor, Dr Daryl Essam for his assistance and guidance, and particularly, his invaluable help with the game-of-Go side for this thesis.

I would like to thank my dear colleagues and friends for helping me throughout my way; Theam Foo Ng, Essam Soliman, Saber El-Sayed, Noha Hamza, Ahmed Fathi, Abdelmonaem Fouad, Mai Shouman, Mohamed Mabrok, Mohammed Esmaeil-Zadeh, Eman Sayed, Irman Hermadi, Dominik Beck, Miriam Brandl, Gabriele Lo Tenero, Miriam Glennie, Farhana Naznin, Hafsa Ismail, Ibrahim Radwan, Andrew Asfaganov, Ahmed Thabet, Osama Hassanein, Yasmin Abdelraouf, Ahmed Samir, Ziyuan Li, Helen Gilory, Evgeny Mironov, Camille Pécastaings, Federica Salvetti, Sheila Tobing, Toby Boyson, Luis Espinosa, Bruno Barbosa, and my best mate Nizami Jafarov.

I would also like to thank all my fellows in the *Computational Intelligence, Modelling, and Cognition Lab. (CIMC)*. I particularly would like to deeply thank: Shir Li Wang, Wenjing Zhao, Van Viet Pham, Murad Hossain, Bing Wang, Heba El-Fiqi, George Leu, Kun (Liza) Wang, Sondoss El-Sawah, Shen Ren, Jiangjun Tang, Weicai Zhong, Sameer Alam, Bin Zhang, James Whitacre, Vinh Bui, Lam Bui, and Kamran Shafi.

I am very grateful to Dr Sherene Suchy; her guidance, support and friendship have made it possible for me to achieve my potential. I am also grateful to many members of the school, who provided such a wonderful environment, especially, Ms Elvira Berra, Mrs Elizabeth Carey, Ms Christa Cordes, Mr Luke Garner, and Ms Mette Granvik.

I am also very grateful to my many friends in Egypt, who proved to me how true friends can grow separately without growing apart. As always, you kept me going on guys!!

Finally, for their tireless support and encouragement, I would like to thank my family – *My Mom, Dad, sister Ghada, and brother Ayman* – for providing me with their love and support throughout my PhD.

[This Page is Intentionally Left Blank]

Originality Statement

I hereby declare that this submission is my own work and to the best of my knowledge it contains no materials previously published or written by another person, or substantial proportions of material which have been accepted for the award of any other degree or diploma at UNSW or any other educational institution, except where due acknowledgement is made in the thesis. Any contribution made to the research by others, with whom I have worked at UNSW or elsewhere, is explicitly acknowledged in the thesis.

I also declare that the intellectual content of this thesis is the product of my own work, except to the extent that assistance from others in the project's design and conception or in style, presentation and linguistic expression is acknowledged.

Amr S. Ghoneim

Signed ……..

Date ……..

[This Page is Intentionally Left Blank]

Copyright Statement

[This Page is Intentionally Left Blank]

List of Publications

Ghoneim, A. S., D. L. Essam, and H. A. Abbass (2011). Competency Awareness in Strategic Decision Making. In Proceedings of the IEEE First International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), pp. 106–109, 22–24 Feb. 2011, Miami Beach, FL, USA. IEEE Press.

Ghoneim, A. S., D. L. Essam, and H. A. Abbass (2011). On Computations and Strategies for Real and Artificial Systems. In Advances in Artificial Life, ECAL 2011: Proceedings of the Eleventh European Conference on the Synthesis and Simulation of Living Systems, edited by Tom Lenaerts, Mario Giacobini, Hugues Bersini, Paul Bourgine, Marco Dorigo and René Doursat, pp. 260–267, 8–12 August 2011, Paris, France. MIT Press.

Ghoneim, A. S., and D. L. Essam (2012). A Methodology for Revealing and Monitoring the Strategies Played by Neural Networks in Mind Games. Accepted by the 2012 International Joint Conference on Neural Networks (IJCNN), at the 2012 IEEE World Congress on Computational Intelligence (WCCI 2012), Brisbane – Australia, June 10-15.

[This Page is Intentionally Left Blank]

The Knowledge of anything, since all things have causes,

is not acquired or complete unless it is known by its causes.

--- Avicenna (980-1037)

[This Page is Intentionally Left Blank]

Contents

List of Figures

List of Tables

[This Page is Intentionally Left Blank]

Chapter 1: Introduction

The key to growth is the introduction of higher dimensions of consciousness into our awareness.
–––Laozi (6th century BCE)

1.1 Overview

The process of Decision-Making has been widely studied across many different domains and perspectives. Decision-Making, which is practised by humans day after day, can be simply defined as the process of selecting a final action among several available alternatives (Raiffa, 1968; von Winterfeldt and Edwards, 1986). The intention is thereby to achieve a "more desirable" targeted situation given the current state of the system. Yet, to decide upon that action, this cognitive process involves numerous activities and skills; including: perceiving and recognizing the current situation, determining and prioritizing the desired objectives, as well as generating and evaluating the alternative actions. The complexity of the activities involved varies significantly, in accordance to the complexity of the situation to be managed, and the proficiency of the practised skills is derived from the experience of the decision-maker.

In the real world, the decision-maker is usually confronted by complex scenarios; including, for instance, dynamic environments in which the decision-maker constantly – and perhaps rapidly – needs to adapt-to and consider a new situation. Complex scenarios also include environments with a large number of aspects that must be taken into account, environments with a large number of potential states that need to be evaluated, or environments that are difficult to define. Along with such environments, decision-makers practically need to make decisions under increasing time-pressure, uncertainty, intricacy, risk, and ambiguity. Yet, the complexity of the environment is only an element of the overall complexity, which contains also the structural complexity of the decision-making process itself (Bullen and Sacks, 2003).

This later complexity has been investigated from many perspectives, such as the multidisciplinary field of human factors (Cook et al., 2007).

Complex situations are context dependant, and as a result, decision-makers need to achieve an awareness based on the current situation in order to develop the decisions that realize their desired objectives (Albers, 2005), (Padilla et al., 2007). Constructing this mental representation of a situation is fundamentally based on how the decision-maker perceives the context. Therefore, the decision maker needs to maintain a complete and coherent situation representation, which is based on relevant existing knowledge, and is updated continuously with new information extracted from the environment. The development of this situation representation is coupled with the ability to both project the future states of the situation, and to also anticipate the results of the actions taken.

Unsurprisingly, decision-making in complex situations has been the subject of active research from diverse perspectives. The aim of such research is to examine the many aspects of observed human decision-making along varying real-world scenarios. The ultimate aspiration is to equip decision-makers with the needed tools to perform effectively in complex situations by making decisions which are in their best interest.

In the fifties, the limits of the human cognitive-resources that affect the reasoning of a decision-maker were investigated, giving rise to the term *"Bounded Rationality"* and to also the concept of satisficing as an alternative to optimizing (Simon, 1956). In the late nineties, the synthetic field of neuroeconomics (Glimcher et al., 2009) emerged in an attempt to explain human decision-making in light of the combined advances in – amongst others – neuroscience, behavioural economics, and cognitive psychology. Recent studies include, for instance, supporting the making of interconnected decisions that span multiple perspectives by developing a generic object-oriented-based architecture for decision-making (Liew and Sundaram, 2005), outlining the potential progress in our understanding of Social Decision-Making as a result of coupling models of neuroeconomics and game-theory (Sanfey, 2007), and putting

forward a framework to explore the different aspects of decision-making from a neurobiological perspective (Rangel et al., 2008).

At another level, a principal way of supporting decision-makers in this difficult cognitive process is by automating the complex system. As part of this, Artificial Intelligence (AI) has progressively been supporting decision-makers through a range of techniques (Phillips-Wren and Jain, 2006), including Machine Perception, Knowledge Acquisition and Representation, Reasoning, Planning, Logic, Computational Intelligence, Multi-Agent Systems, Cognitive Architectures, and Machine Learning. Through these techniques, AI supported the decision-making process by tackling the complexities associated with real-world scenarios. Namely, through simulating/modelling the considered system and investigating its dynamics, perceiving the system states and facilitating up-to-date information, reducing uncertainty and information overload, generating and exploring actions, and so on.

Positioned at the core of AI, Computer *Game-Playing* (GP) provides an exceptional venue to develop automated agents that mimic – and frequently surpass – human performance. That is because GP addresses fundamental questions, starting with search, learning, and knowledge representation and continuing towards planning, multi-agent systems and opponent modelling. Board games specifically are abstract and relatively easier to represent, in addition, they possess straightforward limited rules that can be clearly applied (Luger and Stubblefield, 1998). Since its inception, GP explicitly aimed at providing suitably demanding environments for developing and testing techniques that are useful in attacking other problems of similar nature elsewhere (Shannon, 1950).

Nevertheless, the role of games has always been manifold. An obvious aspect is in the increasing popularity of GP as an influential tool for learning and education (Malone, 1981; Akilli, 2011), as seen in the examples of (Squire, 2006; Tan and Biswas, 2007; Iacovides, 2009; Pavlas et al., 2010; Clerveaux et al., 2010). However, a significant and influential role was – and continues to be – in attempting to understand

and reproduce human intelligence, or what is called *Computational Psychology*. Together with *Computational Philosophy* – i.e., producing the general superior form of computer-based intelligence – they represent the two principal motivations of AI (Shapiro, 1992; Ernandes, 2005). This capacity to explain human intelligence is due to the fact that computer games, in general, allow the relating or comparing of the "computational" intelligence of an automated agent to the "biological" intelligence of its human counterpart (Mayer and Maier, 2005). For that reason, GP is progressively considered as a major tool for genuine advances in AI (Ernandes, 2005).

The contributions of games to our understanding of human decision-making through experience were mainly accomplished through Chess, of which a wide spectrum of studies exists in the literature. The evolution of Chess as a major area for studying expertise was started by de Groot's (1946) *"Thought and Choice in Chess"* in which he investigated the processes of decision making and their variances in relation to expertise (Gobet, 2006). This study also pointed out a difference in how chess positions were perceived between experts and amateurs. Studies then moved along, mostly by statistically and interpretatively analysing verbal utterances that were gathered using think-aloud protocols from Chess players while they were deciding on their next move (Newell and Simon, 1972; Gobet, 2006; Connors et al., 2011). In addition, other studies frequently experimented with computer simulations to further understand human problem solving and expertise in chess. Varying psychological topics were investigated; for instance, perceptual and memory processes (Simon and Chase, 1973; Gobet, 2005) and intuition (Frantz, 2003; Linhares, 2005). The topics also include problem solving, decision making, and intelligence, and all of which can be found summarized in recent studies, were also either as investigations from an exclusively Chess perspective (Rasskin-Gutman, 2009), or from a broader board-games perspective (Gobet et al., 2004). Similarly, De Groot's (1946) earlier studies were recently replicated to examine the current arguments concerning expertise and its causal cognitive processes (Connors et al., 2011).

Up to the present time, Chess – and computer game playing in general – has been expected to play a vital role in understanding and explaining human intelligence (Ernandes, 2005; Linhares, 2005; Harré et al., 2011a). However, from the time when a Chess world champion (i.e., Kasparov) was defeated by a computer program (i.e., Deep Blue), the *"research on computer chess seems to have gradually faded to the background"* (Linhares, 2005). As an alternative, researchers have advocated considering other problems to be at the forefront of AI, such as automated story-telling (Bringsjoid, 1998). However, practically, *"Go has emerged as a key testbed for research in [AI]"* (Harré et al., 2011b).

Go belongs to the category of "strategy-games" – which is also called *"mind-games"* (Bouzy and Cazenave, 2001; Mańdziuk, 2007) – in which decision making skills are deeply employed, and where a game's outcome is significantly determined by the players' intellectual capacities; that is to say, victory is won by deeply employing skills such as abstraction, conceptualization, and reasoning. GP in strategy games addresses some vital high-level strategic elements, such as opponent modelling and managing uncertainty.

In general, Strategy board-games – e.g., Chess, Checkers, and Go – are distinguished by their comparatively simple tools; typically a marked-board and movable pieces, uncomplicated predefined set-of-rules, and a strategic objective. The attractiveness and complexity of Go lies in the fact that it has very simple rules, but an extremely large and complex search space. This reasonably uncomplicated set-of-rules allows beginners to easily attain a basic-level of skill, while allowing experts to attain advanced-levels of skill. Nevertheless, Go's reliance on a sequence of actions to generate patterns mimics the characteristics of many real-world problems; as in strategy design in government. The characteristics of Go, and other strategy games, also include the *"generally symmetric nature of play"*, the strategically-equal opening points, the concentration on strategic-decisions to achieve the goals of the game, and those goals which typically involve the *"elimination of other players through simulated military*

conflict" and/or *"[dominating] most of the territories of the game"* (Loh and Soon, 2006; Sharma et al., 2007).

Consequently, Go is both an ideal and a grand challenge to AI (Cai and Wunsch II, 2007; Rimmel et al., 2010; Hoock et al., 2010; Harré et al., 2011a). The differences between Go and other games – including Chess – in complexity measures is shown in (Figure 1.1), with the complexities of Go far larger than that of any of the other perfect-information games. Hence, the complexity and combinatorics of Go have been explored by several studies (Lichtenstein and Sipser, 1980; Müller, 2000; Wolfe, 2002; Tromp and Farnebäck, 2007). The difference between Go and Chess were investigated in (Burmeister and Wiles, 1995) and include – among others – a higher branching factor, an increasingly far-sighted look-ahead, a more significant horizon-effect, and as a result, a more complicated overall board-evaluation. Those differences resulted in the remarkable failure of the classical search-based methods in solving Go. Consequently, computer Go has just achieved a professional level using a smaller 9×9 board (Enzenberger et al., 2010), which is considerably uncomplicated when compared to the standard 19×19 board. However, unlike Chess and other mind-games, there are no Go programs that can challenge professional human players on the standard board (Harré et al., 2011a).

Recently, this grand challenge has been attributed to a fundamental problem; the lack of relevant and practical research on understanding and explaining human intelligence and its development with experience (Ernandes, 2005; Harré et al., 2011a). As stated in (Harré et al., 2011a), the expertise in Go is *"a crucially important indicator of the unique factors of human performance"*. This is the heart of this study; investigating the ability to practically evaluate the competency of Go players. In the upcoming section, we will state the motivation for this work, in which we will introduce the current status of computer Go, its recent trends and shortcomings. Then, we will bring in the necessity for an advanced competency awareness that we see as an inherently essential component of an improved overall situation awareness, and as an approach for further advancement in computer Go. Section 1.3 will lay out the research

question and hypothesis of this thesis, followed by outlining the original contributions of this study (Section 1.4) and the organization of the thesis (Section 1.5).

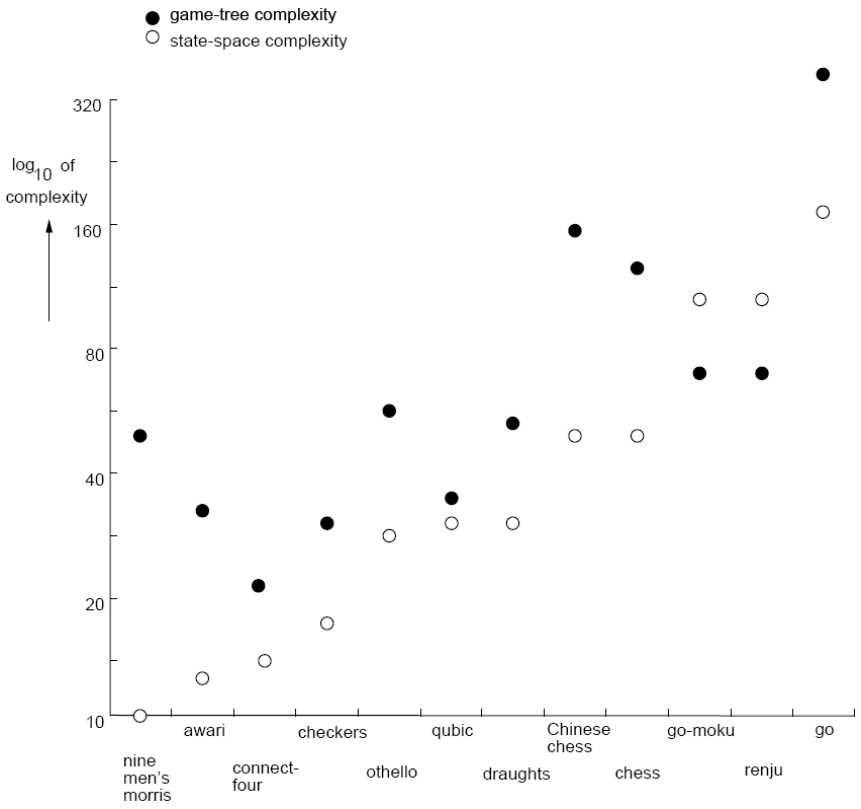


Figure 1.1. Estimated Game Complexities (After Allis, 1994). Game-tree and State-Space complexities are explained in Section 2.2.1. Alternative definitions of a Game-tree complexity, and consequently, of GO's estimated complexity exist; see for example: Tromp and Farnebäck (2007).

1.2 Research Motivation

Developments in computer Go can be generally characterized from two perspectives. From a computational perspective, computer Go can be considered on the basis of the approach employed, however – frequently – the resulting categories overlap in practice. The main approaches are:

- **Knowledge-Based (KB) Approaches;** are among the first techniques employed in Computer Go, and in which domain knowledge is explicitly

integrated into computer players (i.e., the game engines). The knowledge is typically in the form of patterns' datasets (Abramson and Wechsler, 2003), as well as hand-tuned rules (Bouzy, 1996) that represent the varying useful concepts in the game. Independently, Knowledge-based approaches have so far been fairly successful. Moreover, domain-based patterns and rules constitute the typical means of integrating knowledge into Machine-Learning or Monte-Carlo based approaches.

- **Machine-Learning (ML) Approaches;** are techniques of knowledge-discovery in which the Go knowledge is not built-in by hand. These mostly use Neural Networks (Sutskever and Nair, 2008; Wu and Baldi, 2008) or Neuro-Evolution (Mayer and Maier, 2005; Cai and Wunsch II, 2007). ML-based approaches have so far achieved insubstantial outcomes, and – consequently – have received less attention than their Monte-Carlo based counterparts.

- **Monte-Carlo (MC) Approaches;** are approaches based on randomly exploring the search-space, and are – by far – the most successful basis for developing computer Go players (Gelly and Silver, 2008; Lee et al., 2010a; Rimmel et al., 2010). Similarly to ML-based approaches, no domain knowledge is directly integrated in the standard MC approach. However, practically, domain knowledge is increasingly being incorporated (Hoock et al., 2010).

Besides the abovementioned categories, alternative less-popular approaches have been employed, including – but not limited to – adversarial planning and reasoning (Klinger, 2001; Willmott et al., 2001), combinatorial game theory (Müller, 1999), Markov networks (Raiko, 2005), and mathematical morphology (Bouzy, 2003).

An alternative, less obvious perspective, is to focus on specific sub-problems in computer Go in order to reduce the overall complexity. This is achieved by breaking-down the game into sub-games (Müller, 2003), the board into local areas (Stanley and

Miikkulainen, 2004), or the overall plan into individual strategies (Cai and Wunsch II, 2007). As a result, those partial problems become sufficiently simpler to be tackled directly.

In spite of recent advances in computer Go programs, which are now able to challenge professional players, substantial improvements are still required. Additionally, Go's unusually complicated evaluation function remains a dilemma to encode, in spite of its simple rules, and in spite of the huge compilation of information that exists in literature and that covers in details the various strategies and tactics employed through it. The difficulty in encoding the evaluation function restricted the rate of improvements in the knowledge-based approaches; on the other hand, it endorsed the utilization of – the domain-independent – Machine Learning methods and Monte-Carlo simulations to learn and/or estimate this function. However, ML approaches are becoming stalled by the credit assignment problem. This is due to uncertainties – even among the most experienced players (Hoock et al., 2010) – in evaluating a best course of action, and the fact that feedback – in virtually all the automated systems – is received for the entire game rather than for each decision. Methods for indirect/implicit learning, such as evolutionary algorithms, were thus increasingly utilized. However, a lot of issues have arisen due to the inability to understand the nature of the learning process; in terms of: what type of play is being evolved, what is the effect an opponent would have on the evolved strategies, how to guide the evolution of higher-quality players, and more importantly, whether or not did the evolution process stagnate. These issues pose severe limitations to advancing ML-based Go players.

The application of Monte-Carlo tree search (MCTS) in Go, regardless of its unquestionable success, is also caught up by the domain-independent nature of the approach, and its lack of learning. MCTS in Go suffers, for instance, from many tactical limitations (Bouzy, 2005; Chaslot et al., 2010; Hoock et al., 2010; Rimmel et al., 2010). Lately, most research has progressively – and inescapably – incorporated expert-knowledge; generally in terms of patterns and rules (Bouzy, 2005; Gelly et al., 2006),

knowledge-based heuristics (Chaslot et al., 2008), reinforcement learning (Gelly and Silver, 2007), and most recently, in terms of ontology-based fuzzy inference (Lee et al., 2010b).

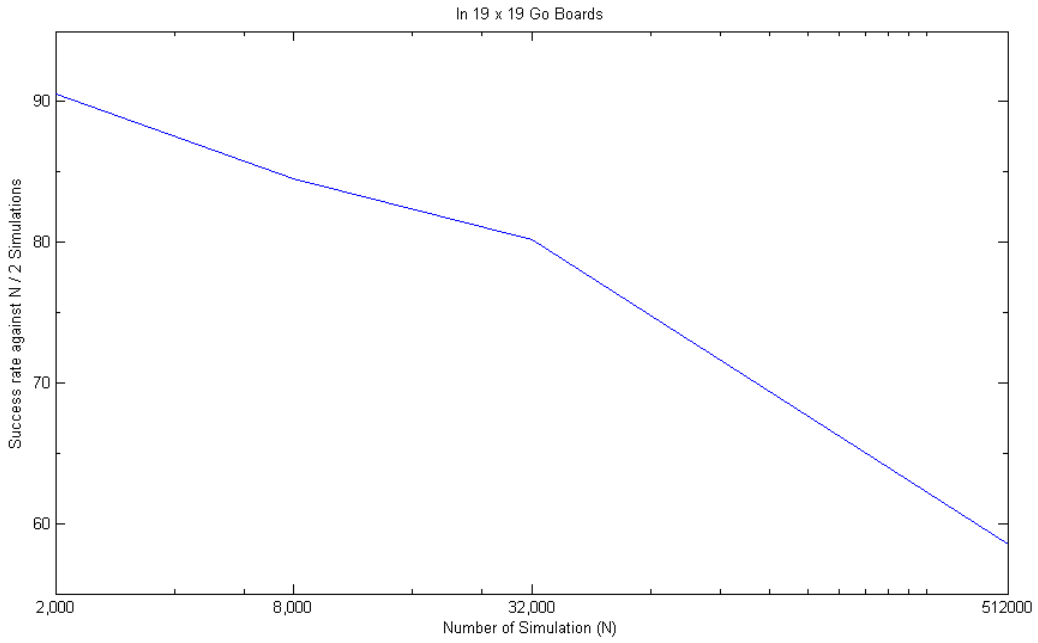


Figure 1.2. Scalability of MCTS for the Game of Go – based on tabled estimations in Hoock et al. (2010).

Across the above-mentioned approaches, we can observe two issues that deter the development of computer Go; namely, a *knowledge-acquisition bottleneck* (Wagner, 2006; Hoppenbrouwers et al., 2009) and a *training bottleneck* (Hoffman, 1998; Ross, 2006). Despite the available domain information, evidently some – and perhaps much – of the knowledge about human-specific methods are not properly acquired. Mańdziuk (2008) listed several of those methods; autonomous learning, pattern-based learning, intuition, knowledge discovery, abstraction, generalization, efficient position estimation … etc. Albeit such methods – which are fundamental characteristics of expertise – might be describable by Go experts, the challenges of acquiring this knowledge are yet to be overcome.

However, a much more significant issue is a training bottleneck. That is, of first highlighting the limitations in the so far developed expertise, and of then secondly adding the necessary knowledge to overcome those limitations (Hoffman, 1998). In other words, *"what matters is not experience per se but "effortful study," which entails continually tackling challenges that lie just beyond one's competence"* (Ross, 2006). This training bottleneck can be seen – to a degree – as a consequence of the inefficiency in extracting Go expertise; i.e., the knowledge-acquisition bottleneck, but given the considerable amount of Go information already extracted from experts, the training bottleneck is mainly the result of highly insufficient competency awareness. This ability to evaluate the skills and competency levels of a decision-maker is an inherently necessary component in the construction of situational awareness (Endsley and Garland, 2000; Banbury and Tremblay, 2004), which is in turn an indispensable foundation for managing complex situations (Jakobson et al., 2007).

In general, recent studies have expressed a similar overall view; Harré et al. (2011b) stated that *"Artificial intelligence (AI) research is fast approaching, or perhaps has already reached, a bottleneck whereby further advancement towards practical human-like reasoning in complex tasks needs further quantified input from large studies of human decision-making."* Studies on cognition and expertise in Go – which to date has received far less consideration than chess – are picking up, and in domains as far as studies that are devoted to assessing the suitability of Go as a research domain for Cognitive Sciences (Burmeister, 2000). In *"The development of human expertise in a complex environment"*, Harré et al. (2011a) utilized tools of information theory to understand and describe the growth of expertise and the associated changes among skilled-amateur and professional Go players. The authors achieved that by quantitatively exploring the uncertainties a Go player has in relation to their opponent's following move. Progress has also been made by focusing on: the response times (Reitman, 1976), the utterances of Go terms *"i.e., verbal reports"* (Yoshikawa et al., 1999), the memory performance (Burmeister et al., 2000), and the human computer interaction (Kim et al., 2009).

Finally, this research shares the same inspiration as various recent studies; that is of, understanding and mimicking the human approach to game playing, and decision-making in general (Ernandes, 2005; Mańdziuk, 2008). As well as potentially affecting the future direction in other complex domains that share characteristics with the game of Go; such as Linguistic, War simulation, Social Sciences, and Economy (Bouzy and Cazenave, 1997).

1.3 Research Question and Hypothesis

Complex situations are very much context dependent. Complexity in this context can be defined as *"the condition of a system, situation, or organization that is integrated with some degree of order, but has too many elements and relationships to understand in simple analytic or logical ways."* (Bennet and Bennet, 2008). Awareness of an agent's competencies, which is principally the result of expertise and training, is imperative for the development of effective decision-making. As a matter of consistent terminology, we would define this state of Competency Awareness as:

> ***Definition 1: the perception and understanding of the set of knowledge, skills, and characteristics needed to allow an agent to perform a specific task with high performance.***

Recent studies in the complex game of Go advocates – for any further improvements – the need for effectively discovering and understanding the exceptional features of human expertise and proficiency. However, currently, the methods described in the literature fail to measure the competency from an objective view, either by depending on manual approaches (Mirabile, 1997; Pew, 2000), or by using automated approaches that depend entirely on the degree to which the goals are achieved (Ghoneim et al., 2011a); i.e. the final outcome/score. The difference between manual and automated approaches can be précised as: *"[the former approach] depends on the construction of models by hand (aided by tools), by individuals or teams [while an automated approach will use] knowledge discovery from data and machine learning techniques to derive models"* (Hoppenbrouwers and Lucas, 2009). These are the

principal arguments of this research, or, in a more formal way, this study attempts to address the following question:

Can the Competency Level of an agent within the complex situation of a Go game be assessed automatically and objectively, and if yes, can this assessment be used to guide learning in computer players?

In order to answer this question, it is hypothesized that the state of competency awareness can be attained or enhanced by developing an objective/quantitative assessment; that will act as a well-accepted standard, and against which performance can be effectively measured, and – consequently – by which training and development can be effectively guided. In addition, it is also hypothesized that criteria, such as Situation Reasoning – *Which can be defined as the processes of perceiving and understanding a situation, and the outcome of such processes as explanations and justifications (Tilley, 2004),* correlates with the performance in Go, and thus, if satisfactorily estimated, can provide us with the desired competency assessment. However, to methodically investigate the hypothesis and address the research question, we need to address the following sub-questions:

1. **How can a selected competency get evaluated objectively, quantitatively, and passively?**

   The current approaches to evaluate competencies have numerous disadvantages; such as the lengthy data-collection processes that are time-consuming, invasive and subjective in nature. Therefore, it is important to address those drawbacks in any proposed assessment framework.

2. **How does the selected competency assess – and consequently, characterize the development of expertise in human Go players?**

   An improved understanding of the nature of expertise in Go has been emphasized by recent studies (Burmeister, 2000; Harré et al., 2011a). Therefore,

the results of an assessment need to serve as baseline, to which classical Go games – played by human players on 19 by 19 boards – can be measured.

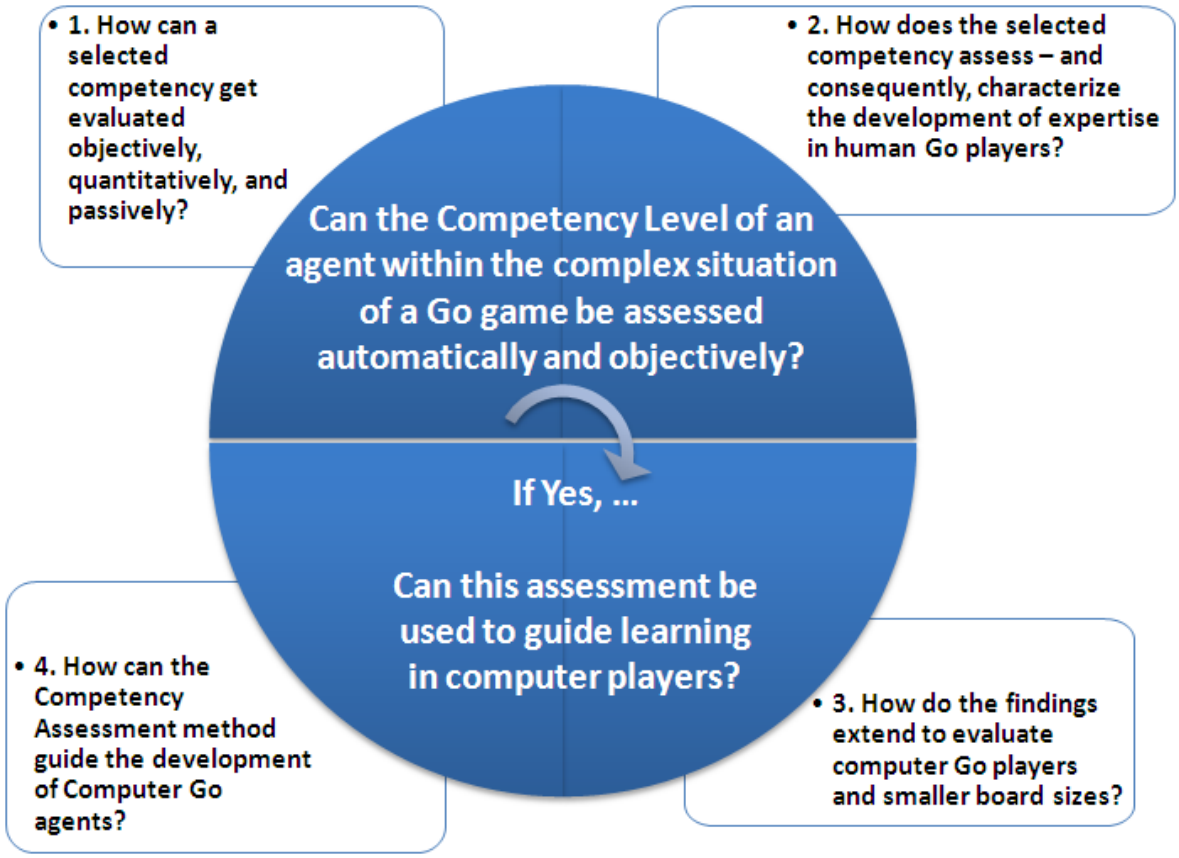


Figure 1.3. Summary of Research Question and Sub-Questions.

3. **How do the findings extend to evaluate computer Go players and smaller board sizes?**

   Computer Go is a grand challenge to AI. Consequently, many of the computerized approaches utilize smaller board sizes because of the demanding computational effort associated with the standard 19 by 19 boards. It is thus crucial to investigate the findings in the light of games played by computer agents and using smaller board sizes.

4. **How can the Competency Assessment method guide the development of Computer Go agents?**

   An objective of the improved assessment will be to better guide the development and training of Go players, and specifically the computerized players. Therefore, it is essential to investigate the effects of employing the devised assessment method on developing Go agents.

1.4 Original Contributions

The main contribution of this thesis is to enforce the importance of developing automated methods for competency assessment that effectively improves the overall situational awareness. Such an objective quantitative assessment method will allow a more enhanced training method for computer Go, and an opportunity to improve our understanding of expertise in such a complex scenario. In more detail, the Original Contributions of the thesis can be listed as follows:

- Presenting a passive, fully computerized methodology to examine how the player's expertise is applied to a complex scenario, and to the skills and competency level of the strategic decision maker.
- Showing quantitatively how human strategies employed during Go games can be categorized, and how those strategies evolve with experience. In addition to exploring the ability to automatically monitor a human/computer player's skill and competency level.
- Presenting a methodology to semantically unfold the evolution of Go Neuro-Players, and to monitor the overall dynamics of the evolution process.
- Presenting a methodology to guide the evolution of Go Neuro-Players, so as to lead to a better convergence.

1.5 Organization of the Thesis

The thesis is organized in 7 chapters: In Chapter 1, the introduction to the thesis is presented, including an overview of the domain, the motivations to carry out this

research, and the research question, hypothesis and sub-questions. In addition, this chapter lists the original contributions of this study, and concludes with this outline.

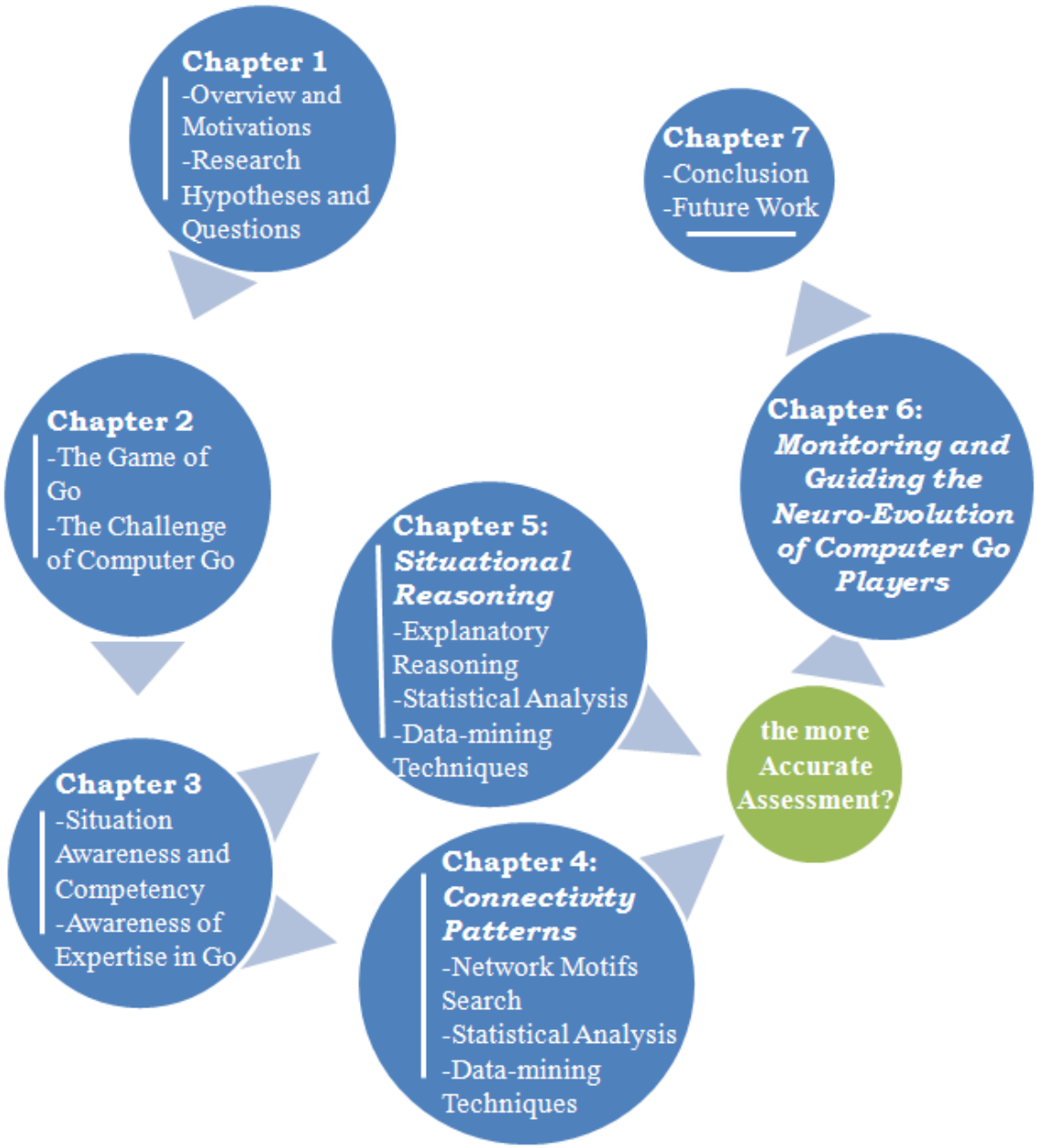


Figure 1.4. Thesis Organization.

Chapter 2 introduces much of the necessary knowledge about the game of Go, followed by a timely review of the current status and the approaches in Computer Go.

In Chapter 3, the issues within computer Go are formalized, and a review of Competency, and its role in complex situation awareness and management is introduced. The chapter concludes by introducing connectivity-patterns and situational reasoning.

Chapter 4 addresses the first and second sub-questions, by proposing a framework whereby a computational environment is used to study and assess the competency of a decision maker in Go. To this end, hundreds of human-played GO games are analysed in a computational environment. A combination of data mining and time-series analysis is developed to monitor and track the competency level of the human players. We then demonstrate that this methodology is successful in diagnosing problems for some players. Chapter 5 readdresses the same questions as in Chapter 4, by considering a modification to the proposed framework, in which an alternative approach to assess the competency of a decision-maker in Go is investigated.

Chapter 6 addresses the third and fourth sub-questions. Firstly, by extending our proposed framework to address one of the disadvantages in neuro-evolving Go Players; by unfolding and monitoring the overall evolutionary dynamics of the neuro-evolution; therefore, observing any stagnation in the evolutionary process, and extracting the semantics of a neural networks' behaviour. Secondly, by investigating the utilization of the approach for monitoring neuro-evolution to guide the process of neuro-evolution; therefore, affecting the development of neuro-Go players. Finally, the thesis is concluded in Chapter 7 by discussing the consequences and implications of the findings, and potential directions for future research.

[This Page is Intentionally Left Blank]

Chapter 2: Computer Go – A Literature Review

This chapter starts by introducing background knowledge about the game of Go, followed in Section 2.2 by a literature review of the varying approaches used in Computer-Go.

2.1 The Game of Go: A Complex Situation

> *A game of Go is symbolic of the gradual occupation of our planet by the human race.*
> *–––Count Daniele Pecorini and Tong Shu,*
> *The Game of Wei-Chi (1929)*

This section introduces the history and characteristics of the game of Go, followed by the rules and a typical sequence of the game. The section establishes the Go terminologies and concepts necessary for understanding the remaining chapters. For a good straightforward introduction to go, the reader may refer to (Chikun, 1997).

2.1.1 History and Characteristics

The traditional oriental game of Go – *Weichi* / *Weiqi* in Chinese, *Baduk* in Korean, *Igo* in Japanese – is one of the oldest strategic board games in the world, and is also one of the most popular. It's generally agreed that it originated in China and is at least *3,000*, and maybe as much as *4,000* years old (Chikun, 1997). Although the game is hard, the rules of the game are few and simple, easy to learn, and flexible enough to accommodate any board size, as well as the standard *19×19* board. Consequently, it is common to introduce beginners to a *9×9* board, instead of the standard *19×19* board; this greatly reduces learning time, because it's less complicated strategically, while it still applying the same rules and tactics.

This game of two players, who simply alternate placing stones on the intersections of the board, is theoretically in the same category as Chess. This is because both games are intellectually stimulating, require high-level strategic thinking, and give chances for players to apply their tactical skills (Chikun, 1997). In addition, Professional players in one of them are usually strong players in the other. Differences between Go and Chess include the initial state, game objective *"[and] thus consequently, how the game ends"*, object values, playing time and requirements, the status of the moves by the end of the game, and the handicap system. Also, a game's complexity arises overtime while the players add more stones to the Go board, while on the other hand it gets less complicated while pieces are removed from the Chess board (Lai, 2004). The game objective also significantly affects evaluation functions in both games. Chess's sole objective of capturing the opponent's king is very clear when compared to Go's mutually dependent objectives of securing your own territory while capturing the opponent's stones (Mayer, 2007). In terms of game variables (Ernandes, 2005), Go can be summarized as a game of 2-players that is:

- *Deterministic*; doesn't include any probabilistic events, thus a current board position is the sole result of the opponents' previous choices.
- *Perfect Information*; Both opponents have an inclusive view of the situation.
- *Discrete in both Time-Flow and Space*; In terms of time-flow, moves are instantaneous and cannot overlap. In terms of space, the board represents an abstract space with a finite number of possible states.
- *Static Environment*; the environment doesn't change during the decision-making phase.
- *Diachronic*; opponents play independently, one at a time.

- *Sequential*; previous choices (moves) directly affect the current and future situations.
- *Closed-World*; the knowledge required to play is finite and clearly defined.

A game of Go starts on an empty *board*; a grid of intersecting lines. The standard *board size* is *19×19*, and yet as indicated above other sizes are common for beginners, as well as for testing computer-Go methods, such as *5×5*, *7×7*, *9×9*, and *13×13*. The game has two players, *Black Player* and *White Player* who alternate place black and white stones, respectively, on the intersections of the board. The *Stones* are coloured round objects, and usually there are *181* stones for black and *180* for white. The players are traditionally ranked using a *kyu* / *dan* rating systems (Figure 2.1). The ranking of the two players in a game affects the *handicap* stones; a number of stones equal to the difference in ranks between the opponents and given to offset this difference. If any handicap stones were given *(normally to the black player)*, then the white player should start the game. Usually, a game starts with a black move, and to offset this advantage of playing first a *komi* – *a compensation value, usually 6.5/7.5 points* – is given. The usual half-point in the komi value is used to avoid a tie.

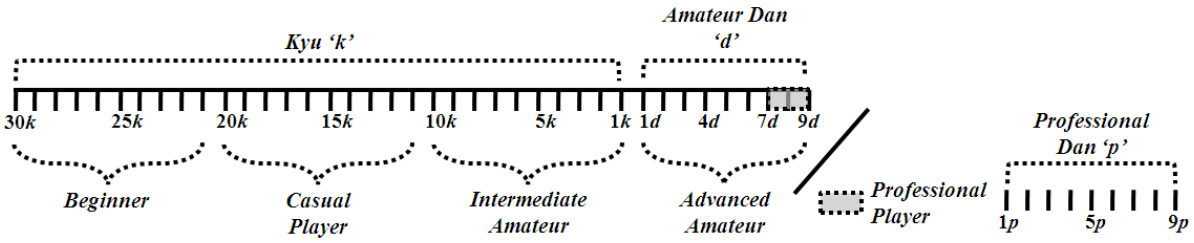


Figure 2.1. The Traditional Go Ranking System.

2.1.2 Rules

The rules of the game are simple (Figure 2.2), the two opponents alternate placing the stones on the intersections of the board with the goal of securing more *territory*; until usually all the intersections that are either surrounded or occupied by the

player. Although, in practice, a player may choose to *pass* his/her turn at any time. In order to achieve the goals of surrounding or occupying intersections, a player connects his/her stones, and thus forms *chains* or *groups*. Connectivity, in this context, is defined as stones of the same colour which are horizontally or vertically adjacent. Diagonal adjacencies and stones of different colours do not form chains. The stronger the group, the more *Influence* (that is, impact) it is has on the neighbouring area, eventually transforming it into a *Moyo*; an area where the influence of the player is large, and thus potentially one which might fall under the control of that player and become part of his/her territory.

Before continuing further, it should be noted that there are a number of variant rule sets for Go. However, the practical effect of these different rule sets is small. Thus although the following describes typical rules of Go, some of the following are rules that are not strictly used in all the rule sets.

A stone is not moved once played, it can only be captured. A stone or a chain is *captured* – *i.e. removed from the board* – iff it has no liberties. A *liberty* is an empty intersection – *free space* – directly adjacent to a stone or a chain, and if all the liberties – of a stone/chain – are occupied by the opponent, that stone/chain is *dead*. Therefore, for a chain to be considered *alive*, it needs to have at least two eyes or the possibility of creating at least two eyes. An *eye* – for a given chain – is an area of connected empty intersections that is entirely surrounded by that chain. A *Suicide* / *Self-Capture* is when a player fills in the last liberty of one of his/her own stones/chains, thus leading to the capture of this own stone/chain. Suicides are usually illegal in most rule-sets.

Another illegal move is any play – *that is not a pass* – that would lead to the recreation of a former board position. This situation typically occurs whenever there is a *ko*; a particular board position in which both opponents may infinitely alternate capturing the same stone. A typical move to re-take a ko is to first play a *ko-threat*; this is a threatening move in a different place on the board – thus altering the board position and tempting the opponent not to fill in the ko – before re-taking the ko.

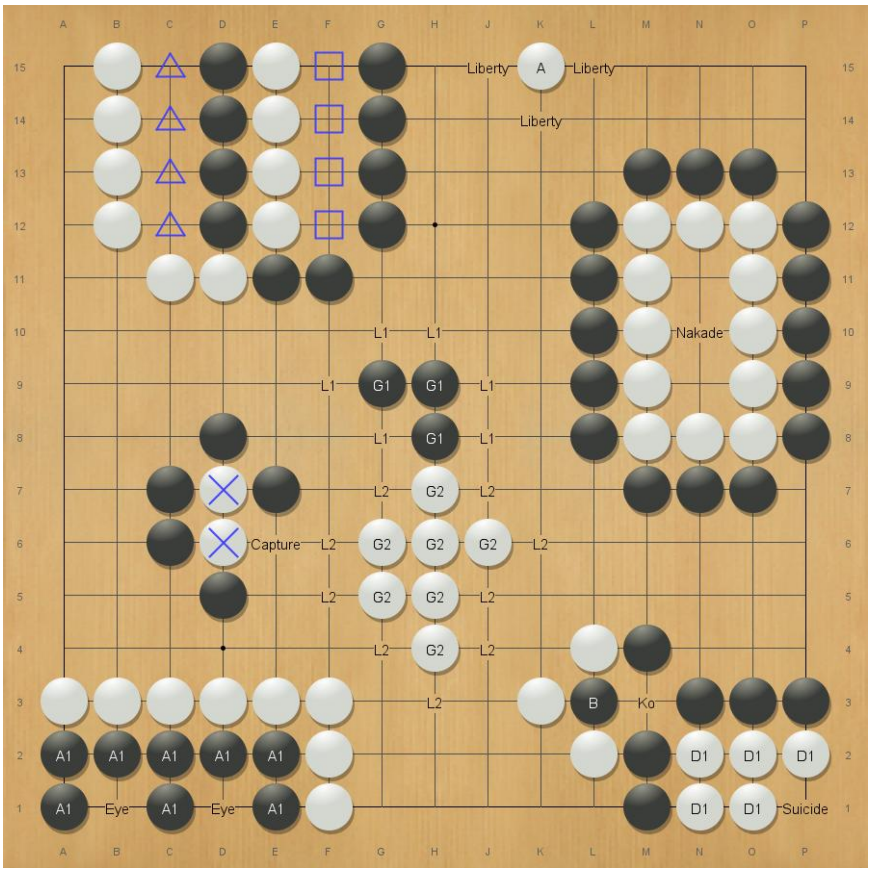


Figure 2.2. The Rules of Go illustrated on a *15×15* *board*. A single white *stone* **(A)** is surrounded by three *liberties* **(Liberty)**. One black *chain*/*group* **(G1)** has six liberties **(L1)** and one white group **(G2)** has eight liberties **(L2)**. If black plays at **(Capture)**, the white group **(X)** loses its final liberty, and thus would be *captured*. If white plays at **(Suicide)**, his/her own group **(D1)** is killed, and would thus be a *suicide*. The white group **(D1)** is *dead* since it surrounds only the single eye **(Suicide)** and so black can capture the group whenever they wish. On the contrary, the bottom-right black group **(A1)** is *alive* since it has two *eyes* **(Eye)** and thus cannot be captured. A typical *life-and-death* situation is *Nakade* where the white group has one large internal space, if black plays at **(Nakade)** white is killed, yet white can live by playing at the same point. At the top-left of the board, a capturing-race, i.e., *Semeai*; Four liberties **(△)** to four liberties **(☐)**, whoever plays first wins this capturing race. Finally, if black just played at **(B)** capturing a white stone at **(ko)**, the *ko* rule applies; white cannot immediately play at **(ko)** and re-take **(B)**.

2.1.3 A Typical Sequence

A typical Go game consists of three phases, an *Opening* (the first 10% to 20% of the game), the *Middle-Game*, and the *End-of-the-Game*. Numerous standard whole-board opening patterns – *Fuseki* (Figure 2.3) – are available, usually starting with a play in a corner, followed by plays on the sides and then at the board's centre. The corner positions are further developed through standard sequences, known as *Joseki* (Figure 2.3). The large Middle-Game phase is spent in *Attacking* the opponent's chains while *Defending* one's chains, two fundamental strategies in Go. Plays that are specific to the End-of-the-Game phase are termed *Yose*. Many of those plays can be represented and learned as *Shapes*; patterns that describe the positional qualities of a group of stones of the same colour while taking into account the nearby opponent's stones. Relevant to shapes is *Suji* – or lines of play – which can be used skilfully (that is, *Tesuji*) or crudely (that is, *Anti-Suji*).

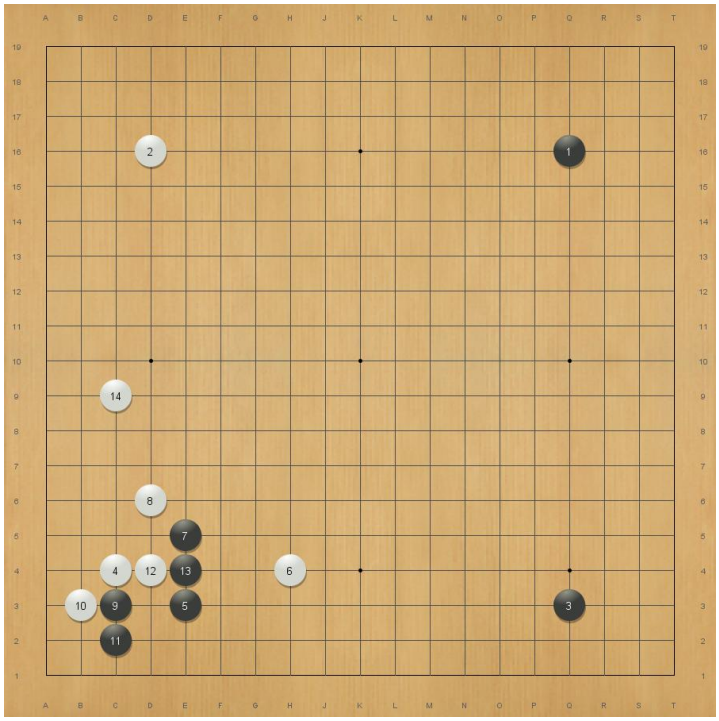


Figure 2.3. The *Opening* of a Go Game on a standard *19×19* board (After Chikun, 1997). A *Fuseki* *(moves from 1 to 14)* including a standard *Joseki* *(the sequence of moves from 7 up to 14).*

A Fundamental concept in Go is *life-and-death*; determining whether the status of a specific group is alive or dead, that is, whether a specific group is going to remain forever on the board, or that it will eventually be captured. Determining in advance whether a group is alive, dead or unsettled is usually termed *Reading*. Specific problems based on life-and-death are termed *Tsumego*.

A game traditionally ends with a consecutive pass from both players. The player with the highest score – *taking the komi into account* – wins the game. Given the fact that Go has slightly varying sets of rules *(e.g., Chinese, Japanese, Korean, … etc.)*, there are two basic scoring systems (Figure 2.4) that mainly result in a slightly varying score value:

- the *Chinese scoring system*; In which a player's score equals the number of empty intersections surrounded by his/her stones in addition to the number of stones he/she has on the board), and …
- the *Japanese (also Korean) scoring system*; In which a player's score equals the number of empty intersections surrounded by his/her stones minus the number of stones he/she lost – by being captured – to the opponent.

2.2 Computer Go

> *Go programming must model human thought itself! Not only must a computer now act like a person, but it must also think like a person! ---Marco Ernandes, AI & Games: Should Computational Psychology be Revalued (2005)*

This section surveys the varying techniques used in Computer Go, and offers the basis for understanding the remaining chapters. The section starts by reviewing the challenges to Computer-Go, followed by the major different approaches to Computer-Go. The reviewed approaches are the Knowledge-Based approaches, the Monte-Carlo

based approaches, and finally the Machine-Learning approaches. After all that, we present in Subsection 2.2.5 some of the assumptions and rule modifications proposed to address the vague parts in the game rules. For more elaborate computer-Go surveys, the reader may refer to (Bouzy and Cazenave, 2001; Müller, 2001; Müller, 2002; Cai and Wunsch II, 2007; Chen et al., 2009).

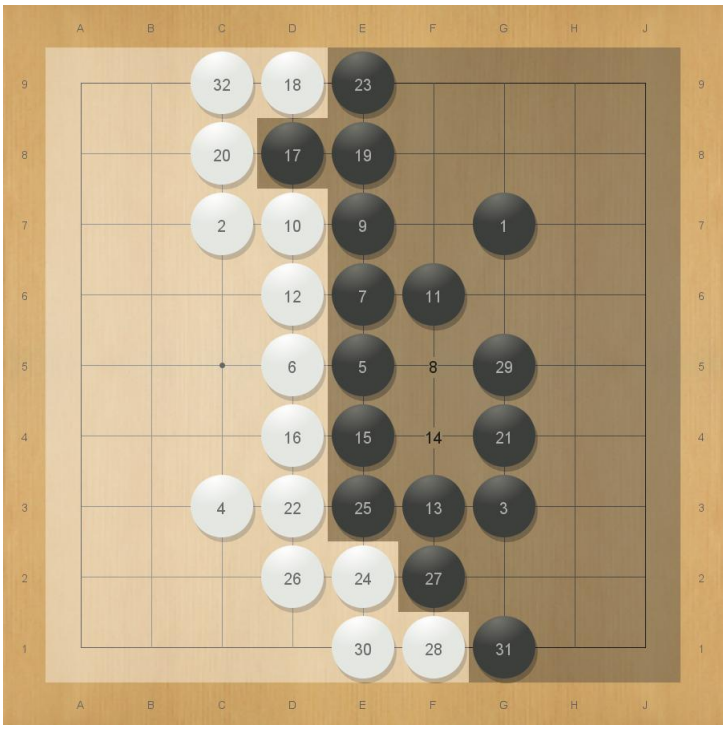


Figure 2.4. Counting the score of a *9×9* game of Go, given that the komi equals *6.5* and that *2* white stones were captured throughout the game. According to the Japanese scoring system, Black has *27* points *(of area)* while White has *28.5 (22 of area plus 6.5 of komi minus 2 captured stones)*, so White wins by *1.5* points. According to the Chinese system, Black has *43* points *(of territory)* while White has *44.5 (38 of territory plus 6.5 of komi)*, so White wins by *1.5* points.

2.2.1 Challenges of Computer-Go

According to (Van der Werf, 2004) the first scientific paper published about the Go game is Remus's (1962) simulation of a learning machine for playing Go. In the 1980's, the beginning of computer Go tournaments, and the release of various computer

Go programs founded this field (Burmeister and Wiles, 1997). Many dissertations were written considering the Game of Go, since as early as Zobrist's (1970) PhD dissertation, who's program was the first to defeat an absolute Go beginner (Van der Werf, 2004).

Computer-Go is very challenging, and due to the structure of the game, a strong computer-Go player *"emphasize[s] much more the human way of thinking"* (Brügmann, 1993), in contrast to the use of brute-force search based methods or other AI techniques that Go remains resistant to. To unravel this ultra-weakly solved game, an engine should think like a human rather than simply acting like one, thus *"[modeling] human thought itself"* (Ernandes, 2005).

Although Go is of the same category, currently available Chess methodologies are not sufficient for, and cannot be translated into the Go domain (Burmeister and Wiles, 1997; Van der Werf, 2004). Therefore, unlike Chess, there are no Go programs that can challenge strong human players in standard games (Enzenberger et al., 2010; Hoock et al., 2010; Tan et al., 2010; Harré et al., 2011a), nor even – *until recently* – moderate human players (Cai and Wunsch II, 2007; Ernandes, 2005). Further, although the *9×9* Go boards have a complexity between that of Chess and Othello (Bouzy and Cazenave, 2001), Computer Go has just reached the top human level (Enzenberger et al., 2010). Therefore, occasionally, simpler versions of Go – that follow the same rules but have different goals – were used while developing a Computer Go engine, such as *Capture-Go* (Konidaris et al., 2002; Cai and Wunsch II, 2007).

With Go as a remarkable exception, search-based methodologies were classically used in solving many games (Van der Werf et al., 2003), including Chess (Hsu, 2002). Searching a game-tree is a typical AI approach that is used to determine which move to play in a two player perfect-information game. The game-tree is a tree of nodes representing the status of various games, where those nodes represent the goal state(s) identified, the arches (edges) represents valid transitions between the nodes (i.e. available moves), in addition to the presence of an evaluation function determining the

value of each node (i.e. state) (Luger and Stubblefield, 1998). Searching the tree can be more efficient using different methods, e.g. the *α-β* pruning method.

Allis (1994) defined the complexity of games in terms of both state-space complexity and game-tree complexity. The former was defined as *"the number of legal game positions reachable from the initial position of the game"*, while game-tree complexity was defined as *"an estimate of the size of a minimax search tree which must be built to solve the game"*. The differences between Go and other games (including Chess) in both complexity measures is obvious as is shown in (Figure 1.1), where the complexities of Go are shown as being far larger than that of any of the other perfect-information games. Allis assumed Go's state-space complexity to be bounded by 10^{172}, and its game-tree complexity *(average branching factor of 250, & average game length of 150 ply)* is nearly 10^{360}, while that of Chess was assumed to be close to 10^{50} and 10^{123} respectively *(an average branching factor of 35, and an average game length of 80 ply)*. In other words, if playing with a very moderate tree depth of *4*, the moves that must be considered by a computer Go program are nearly *10,000* times larger than those evaluated by a Chess program (Mayer and Maier, 2005).

Beside the large branching factor, another unique feature of Go – compared to Chess – namely the more complicated positional evaluation, also hampers any knowledge-based evaluation function. In contrast, the different values of the pieces in Chess, and consequently, the easier evaluation of tactical struggles, makes the overall evaluation of a Chess situation much easier and straightforward when compared to Go (Cai and Wunsch II, 2007).

To the best of our knowledge, the largest published search-based computer Go programs are (Sei and Kawashima, 2000) and (Van der Werf et al., 2003) for boards of size *4×4* and *5×5* respectively. Sei and Kawashima (2000) investigated a search-based program for Go boards of less than or equal to *4×4*, and they showed the resulting game-tree of both the *2×2* and *3×3* boards *(the later is about 500 nodes)*. Van der Werf et al. (2003) improved a standard *α-β* game-tree search for solving the *5×5* board, and

provided preliminary solutions for the *6×6* board. Their program MIGOS *(MIni Go Solver)* also confirmed Sei and Kawashima's *4×4* board solution but in fewer than *700,000* nodes, compared to the *14,000,000* nodes originally required. MIGOS employed a five-goals heuristic function for evaluation; aiming at connecting stones, making eyes, avoiding edge moves, and the two goals of maximizing the number of stones and liberties.

A large search space and a multifaceted positional evaluation are not the sole complexities affecting a computerized Go player, but also the vague parts in the game rules which are difficult to apply. In (2000), Sei and Kawashima have pointed out various problems in the Japanese rule set that need to be dealt with, such as the *'Judgement of Life and Death'*, the *'Definition of End'*, and the *'No Result'* cases. Assumptions and rule modifications are usually made to solve those situations; more details are found in Subsection 2.2.5.

Computer Go methodologies can be roughly divided into two general approaches, knowledge-based methods and knowledge-free methods. The later includes both knowledge-discovery approaches via Machine Learning techniques, and methods that randomly explore the search space, that is, Monte-Carlo based approaches. Another prevalent – yet not distinct – approach is Divide-and-Conquer. Many varying approaches seem to follow Divide-and-Conquer, either by learning one strategy at a time using a Hybrid Evolutionary Algorithm (Cai and Wunsch II, 2007), or by focusing on a limited area of the board at a time using an Artificial Neural Network (Stanley and Miikkulainen, 2004), or by dividing a game into sub-games using Combinatorial Game Theory (Müller, 2003).

2.2.2 Knowledge-Based Methods

Without doubt, one of the most widely discussed of the knowledge-based methods is that of using domain-knowledge encoded into patterns (Figure 2.5), which is the mainstream technique of integrating domain-knowledge into Go engines (Coulom, 2007). Abramson and Wechsler (2003) argued that since Go tactics are rooted in local

move patterns and its strategies based on evaluating a whole board, both (tactical and strategic concerns) are "difficult to integrate in a heuristic evaluation". Thus many computer Go programs are based on databases of patterns and the values of their associated moves. Supporting the argument that Go is spatially defined, is the fact that – unlike Chess for instance – Go is *"spatially structured"* in terms of 1) unmovable homogenous stones, and also the *"spatially defined concepts"* of 2) capture and 3) territory (Harré et al., 2011a).

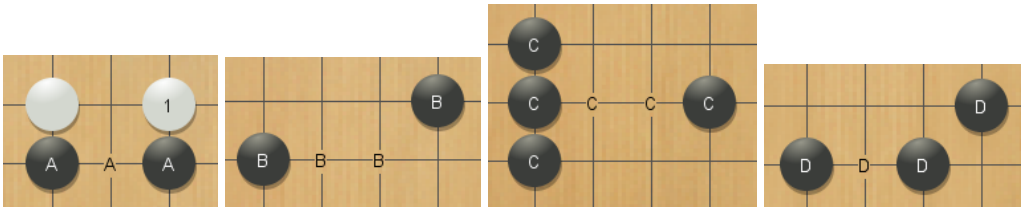


Figure 2.5. Various *Connecting Patterns* for Black (After Müller, 2002).

Bouzy (1996) also agreed that the complexity of Go is due to its spatial properties, and thus it requires the study of human's spatial representation and consequently it depends on spatial reasoning rather than the classical reasoning of "natural language". The spatial reasoning of human players consists of an elementary level where patterns *(of connectors, dividers, or contacts)* are recognized by direct matching, and an iterative level where the previously recognized patterns are iteratively collected into objects *(groups or fractions)*. The whole board is then evaluated by measuring the contribution of the groups in terms of the relationship between a group and its neighbours, and by then using operators such as *closure, in, circling, proximity, and overlap*. A C++ implementation of this cognitive model resulted in the INDIGO Go playing software (Bouzy, 1996).

Most of the domain-dependent knowledge-based Go engines' limitations lie in the weak global sense that results from dividing the whole problem into sub-problems (Bouzy, 2005), which in turn leads to an unusually complex evaluation function where human expertise cannot be straightforwardly encoded, and for which traditional knowledge-based methods do not have the ability to improve. In spite of these

limitations, domain-based patterns and rules are still the mainstream approaches of integrating knowledge, whether into Monte-Carlo based methods or into Machine-Learning methods, both of which – principally – don't incorporate any domain-specific knowledge.

2.2.3 Monte-Carlo Methods

Monte-Carlo Tree Search (MCTS) in Go (Coulom, 2006) – *and generally in other complex scenarios* – aims to tackle the complexity of the evaluation function by replacing it – *i.e., the function* – with a set of simulations that randomly explores the search space. MCTS belongs to the general class of Monte-Carlo (MC) computational methods that rely on repetitive random samplings for computing their results (Figure 2.6). Therefore, traditional MCTS (Figure 2.7) – *and its extensions; such as the Upper-Confidence-Trees (UCT) (Kocsis and Szepesvári, 2006)* – practically uses no expertise or learning. The algorithm depends entirely on performing several random simulations starting from a current situation *s*. For a given *s*, each different possible move *m* is generated and played randomly. Choosing a final move m_{chosen} depends on the proportion of winning simulations; that is, the percentage of games won given that move m_{chosen} was selected as the next move in the simulation.

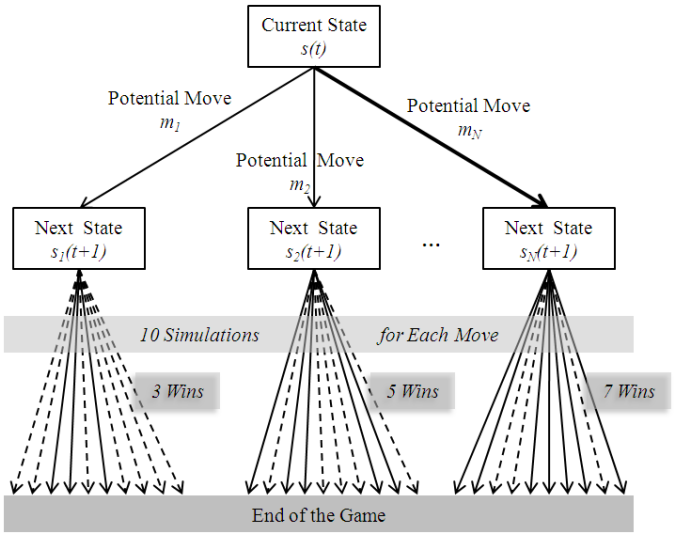


Figure 2.6. Basic Monte-Carlo Move Selection (Based on Coulom, 2009).

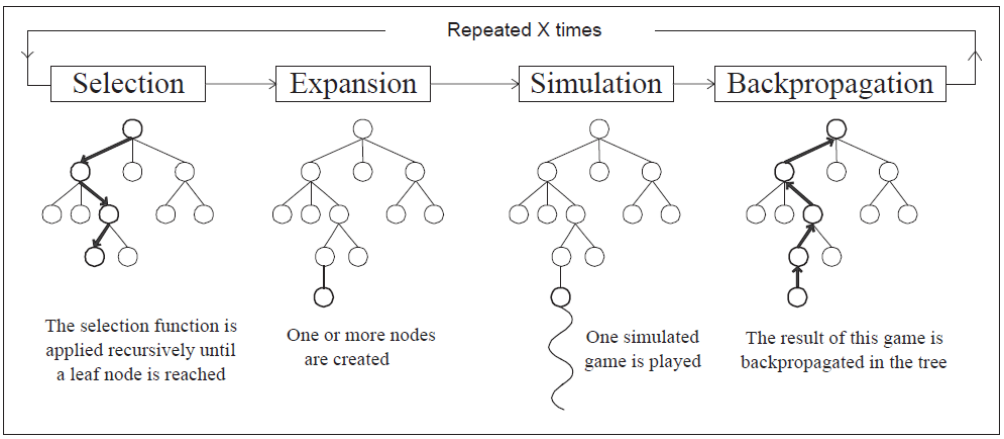


Figure 2.7. Outline of a Monte-Carlo Tree Search (After Chaslot et al., 2008).

MCTS has achieved remarkable results, whether in Go, or in other games such as bridge (Ginsberg, 1999), scrabble (Sheppard, 2006), backgammon (Van Lishout et al., 2007) and poker (Van der Broeck et al., 2009), or in other domains such as planning (Silver and Veness, 2010). In (Brügmann, 1993), the first developed MC-based Go player GOBBLE had a playing strength of about 25 kyu on 9×9 boards. In (Bouzy and Helmstetter, 2003), OLGA and OLEG achieved a level comparable to that of the Knowledge-based engine INDIGO. Currently, MC-based Go engines outperform all other engines, including Knowledge-based engines and engines that employ Machine Learning techniques. Recognized MC-based engines include FUEGO (Enzenberger et al., 2010), Crazy Stone, The Many Faces of Go, and MOGO (Gelly and Silver, 2007). MOGO, for instance, won a 19×19 game against a professional 8 dan in 2008 by 1.5 points and with a handicap of 9 stones (Figure 2.8 (Left)). Soon after, in 2009, MOGO won a game against a professional 9 dan in a 9×9 game (Figure 2.8 (Right)).

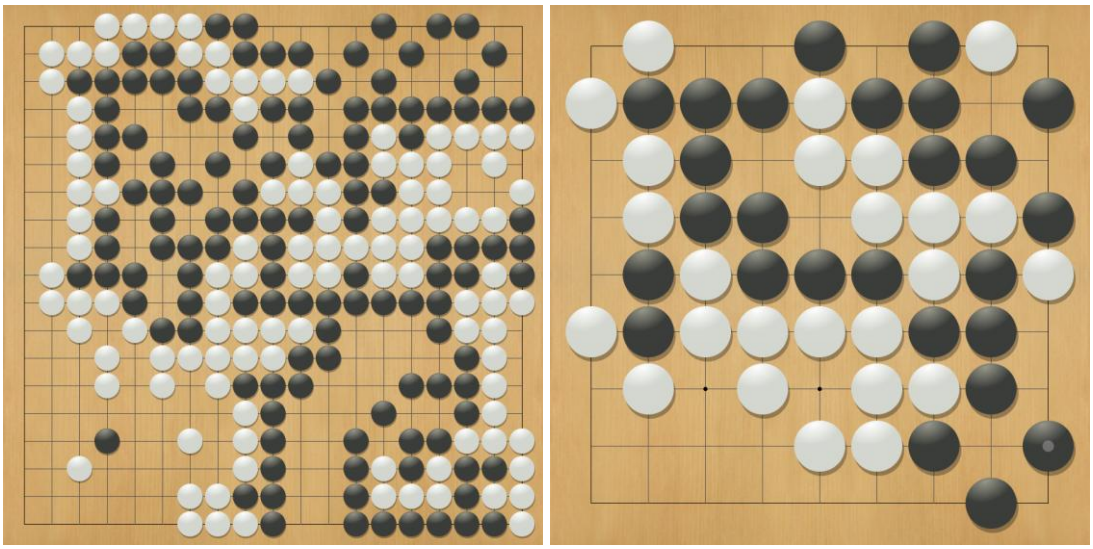


Figure 2.8. The Final Positions of: [Left] The 19×19 Game Won by MoGo (as Black) in 2008 against K.M. Wan (8p), [Right] The 9×9 Game Won by MoGo (as Black) in 2009 against C.H. Chou (9P and winner of the LG Cup 2007).

Yet, despite the undisputed success of MCTS in Go, its *"strengths do not compensate for some big weaknesses when compared to human players"* (Hoock et al., 2010). Those limitations, which result from the domain-independent nature of this approach and its lack of learning, can be summed up as:

- Limitations in Tactical Ability (Bouzy, 2005); since no detailed rules or knowledge can be incorporated or searched for specific local situations.

- Limitations in handling – for instance – some special life-and-death situations such as Nakade (Chaslot et al., 2010); that is the result of MC simulations frequently failing to estimate the unsafe condition of a specific group, and consequently, failing to prevent the death of this group.

- Limitations in Openings (Hoock et al., 2010); due to the fact that MCTS methods cannot learn from experience, and consequently, have no recorded memory of established opening formulations.

Recently, the surprising effects of those limitations, on the scalability of MCTS with increasing computational power, were investigated by Hoock et al. (2010), in which the results have shown a decline in scalability as the number of simulations increased (Table 2.1).

Table 2.1. Scalability of MCTS for the Game of Go (Hoock et al., 2010); showing the effect of increasing the number of MCTS simulations on the success rates.

***N* = Number of Simulations**	**Success Rate of *2N* Simulations Against *N* Simulations In *9×9* Go**	**Success Rate of *2N* Simulations Against *N* Simulations In *19×19* Go**
1,000	71.1 ± 0.1%	90.5 ± 0.3%
4,000	68.7 ± 0.2%	84.5 ± 0.3%
16,000	66.5 ± 0.9%	80.2 ± 0.4%
256,000	61.0 ± 0.2%	58.5 ± 1.7%

Traditionally, several domain-independent improvements have been proposed, such as combining MC-Go with Tree Search (Bouzy, 2004) or implementing a parallel implementation of MC-Go (Yoshimoto et al., 2006). However, recently there seems to have become a general agreement on the inescapability of coupling MCTS with expert knowledge as the way to overcome those limitations. Without doubt, one of the most widely discussed of these is augmenting the MC-based approach with pattern-based domain knowledge.

An attempt to integrate both a MC-based approach and a domain-dependent knowledge-base has been investigated by Bouzy (2005). This supplementary knowledge was used to reduce the poor tactical moves being considered by the MC process, and also to provide it with a more considered evaluation. The type of knowledge that Bouzy (2005) utilized was a 3×3 pattern dataset, in addition to rules about chain capturing. Inspired by Bouzy (2005), Gelly et al. (2006) significantly improved MoGo by introducing 3×3 local patterns to create more reasonable sequences of moves. Those

local patterns were mapped only in the vicinity of the last played move, and thus created some interesting local pattern-sequences (Figure 2.9). MoGo has also been improved in (Gelly and Silver, 2007), by combing the UCT algorithm with value-based reinforcement-learning (Sutton, 1988). In a similar vein, Patterns learned through a supervised-learning algorithm from the game records of strong players significantly improved the MC-based engine Crazy Stone (Coulom, 2007). Alternatively, the MC-based approach has been integrated with *'tactical'* search algorithms which searched for tactical goals; such as creating an eye, connecting/separating two chains, capturing a chain or saving it from being captured, in addition to life and death situations (Cazenave and Helmstetter, 2005).

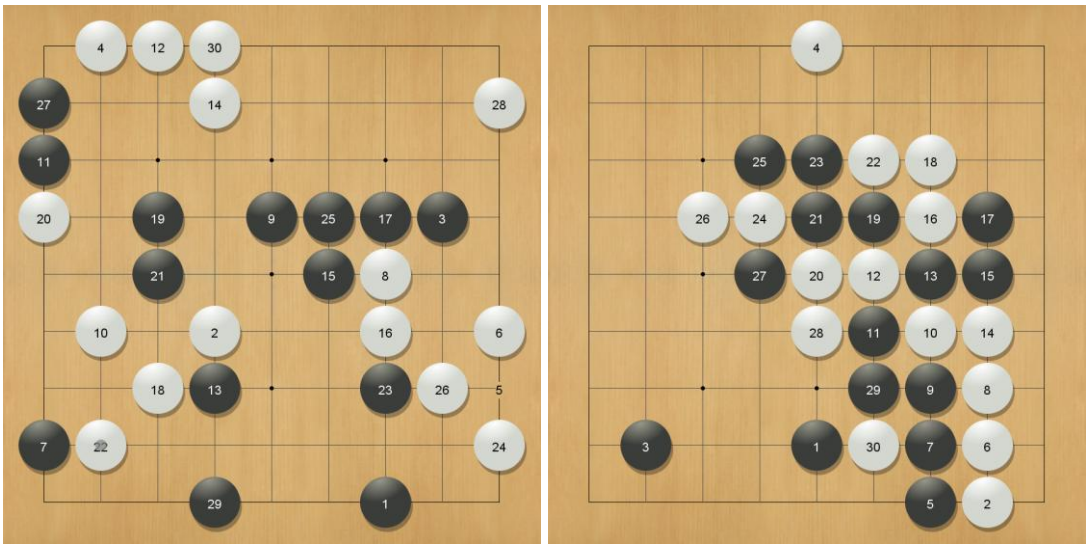


Figure 2.9. The First 30 Moves of One Random Game Simulated by: [Left] Pure Random Mode; *Moves are sporadically played with little sense* [Right] Pattern-based Random Mode; *From moves 5 to 29 one complicated sequence is generated* (After Gelly et al., 2006).

Again, pattern matching (Bouzy and Chaslot, 2005), in addition to rules about *stone(s) capturing/saving* (Bouzy, 2005) and *proximity of the investigated move(s) to the previous move*, were combined to form a knowledge-based Heuristic-value to enhance MCTS (Chaslot et al., 2008). This enhancement noticeably improved Mango's

winning rate against GNUGO on board sizes 9×9, 13×13 and 19×19. In (2010), Chaslot et al. continued incorporating expert knowledge – in terms of patterns and rules – specifically to improve some of the known weaknesses in MC-Go; in particular *life-and-death* situations including *nakade* (Figure 2.10).

The limitations of MCTS were also addressed by Hoock et al. (2010), wherein manual comments by expert players on games lost by MOGO were used to construct a domain-knowledge ontology to represent and – as a result – improve Go openings. The experts' comments in addition to fuzzy-sets were used to build an opening-book that contains improved opening sequences. Besides learning from past experience *(i.e., game records)*, Hoock et al. (2010) also investigated learning in terms of dynamic adaptation during the game by presenting two generic MCTS modifications; *poolRave* and *Contextual Monte-Carlo*.

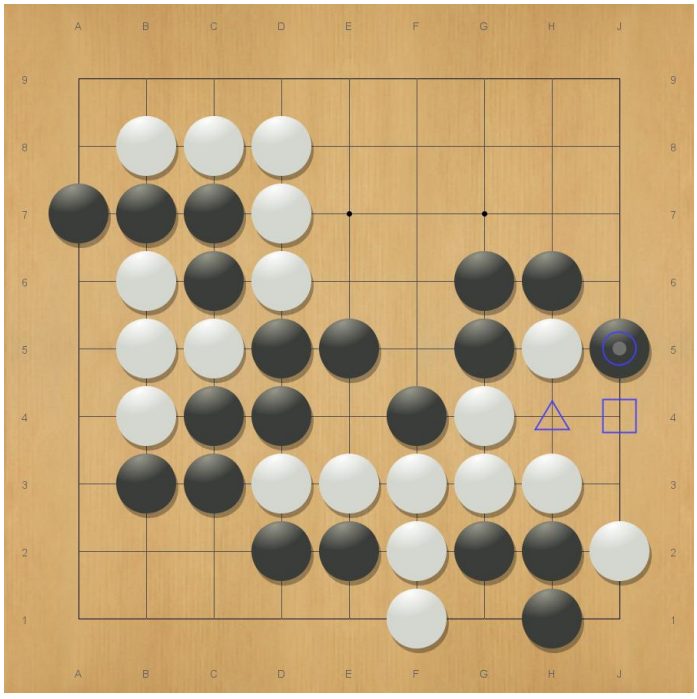


Figure 2.10. A real game played and lost by MOGO; MOGO (white) without specific modification for *nakade* chose **(△)**; black plays **(□)** and the group containing the stone at F1 is dead (MOGO loses). The right move is **(□)**; this move was chosen by MOGO after the modification. (Chaslot et al., 2010)

2.2.4 Machine-Learning Methods

So far, this review of the literature shows that for notable milestones in computer Go – *and other mind-games, such as Chess* – playing is based on the traditional *'brute force'* methods. These methods employ approaches that are definitely not carried out by human players, and ignore certain human aspects, such as autonomous learning and knowledge discovery, and thus have brought about an engineering achievement rather than a scientific one (Linhares, 2005). It has also led to an interest in alternative ways of developing *'thinking machines'* that are able to play based on soft-computing approaches, that is, Machines Learning (ML) and Computational Intelligence (CI) approaches. This idea was expressed by Mańdziuk (Mańdziuk, 2007) who concluded that the final objective of the CI approaches in mind-game playing is the capacity to imitate human behaviour with all its key attributes.

Consequently the techniques required to reach such a level should substitute the currently employed large databases of patterns, rule-based systems and hand tuned heuristics with good capabilities for learning, knowledge acquisition, and pattern-transformation tasks, Neural Networks have always been a practical choice. Neural Networks (NNs) have been effectively applied to various mind-games (Mańdziuk, 2007), and have accomplished outstanding playing in some cases, such as *Neurogammon* (Tesauro, 1989) for playing Backgammon. NNs are practical at pattern recognition tasks, and thus could be highly applicable to games that are largely based on shapes, such as Go. Classically, NNs can be trained to have a board state as an input, and then to map it to another board state that indicates the following move. Though straightforward, this approach is hindered by the credit assignment problem. That is, it is necessary beforehand to know the best move position given any board state. Alternatively, it is important to know which individual moves to reward – *for contributing to a final win* – and which moves – *contributing to a final loss* – to punish. Unfortunately, all that is actually available for some games is simply the final score.

The most widely discussed solution for this problem is Neuro-Evolution; employing an Evolutionary Algorithm (EA) to search for effective networks. NNs are

evolved based on their performance in entire games – *as their fitness* – rather than rewarding/punishing the individual moves. Training NNs by EAs in mind-games has been successfully applied in Checkers (Chellapilla and Fogel, 1999), Chess (Fogel et al., 2004), Othello (Moriarty and Miikkulainen, 1995), and Backgammon (Pollack and Blair, 1998).

Evolving neuro-players to play Go has been implemented in a number of studies. In (1998), Richards et al. evolved two populations, one of neurons and another of blue-prints *"determining which neurons to combine into the network"*, instead of evolving a complete NN (Figure 2.11). This algorithm was known as Symbiotic Adaptive Neuro-Evolution (SANE), and it will be employed in our experiments (see Section 6.2.1). The algorithm was applied, without any pre-programmed knowledge, to small boards; *5×5*, *7×7*, and *9×9*. NNs able to defeat WALLY – *a trivial computer player* – were evolved in *20*, *50*, and *260* generations respectively. Although manually analysing the networks' play revealed that the networks often took advantage of WALLY's weaknesses, the networks also generally displayed some characteristics of common Go playing, such as playing a *Fuseki* near the centre of the board.

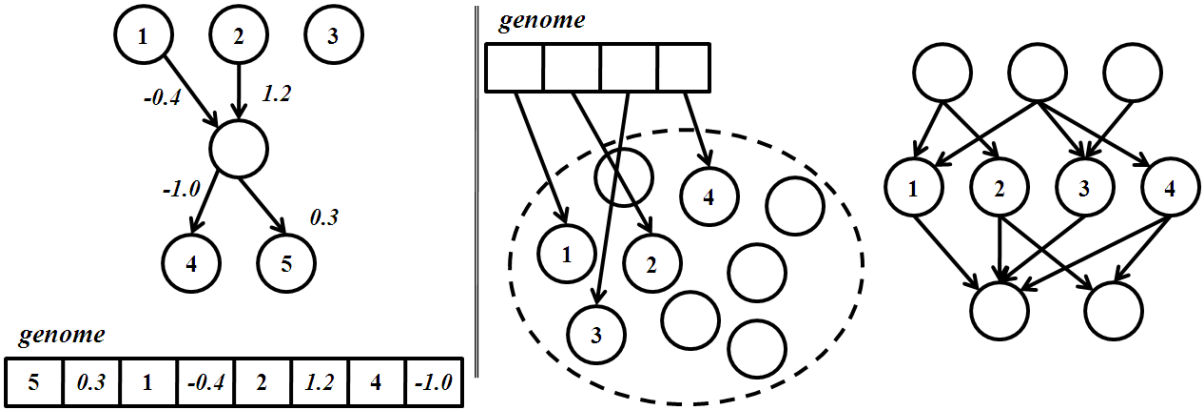


Figure 2.11. Symbiotic Adaptive Neuro-Evolution (After Lubberts and Miikkulainen, 2001). [Left] An individual from the neuron population. The genome (on the left) specifies which connections are to be made and what weight they have. [Right] An individual from the blueprint population. The genome (the bar) points to neurons in the neuron population of which the hidden layer is made up.

Evolving NNs for game playing naturally needs an opponent to evaluate the fitness of those evolved networks. Thus, since they do not rely on human expertise, evolved NNs may have playing behaviours beyond those of human players. However, this indispensable need for the presence of an opponent led to strong disadvantages, especially with the facts that 1) in some games the available computer programs are still immature, and 2) even any well-built opponent(s) will eventually be exhausted. This is a problem because the opponent is the only source of information the network has about the game. As a result, the NNs' need for developing strategies is limited by what is required to defeat the opponent. In addition, the NNs have frequently evolved strategies that exploited the weaknesses of the opponent, instead of showing common high-quality playing capabilities (Richards et al., 1998).

Several solutions have been proposed to overcome the previously mentioned drawbacks of evolving against a computer program:

- Creating a nondeterministic aspect in the opponent's play by adding a random factor (Richards et al., 1998),
- Evolving against varying opponents (Richards et al., 1998), or the more established
- Co-Evolution (Lubberts and Miikkulainen, 2001), (Mayer and Maier, 2005).

In more details, Lubberts and Miikkulainen (2001) co-evolved two populations, one population that searches for optimal solutions and another population of *"test cases"* that challenges the former one *(so-called competitive co-evolution)*. In addition, they used the *Hall-of-Fame* concept by examining the evolved solutions against the best of the previous generations to ensure a rising quality. When tested using 5×5 boards against networks evolved against the GNUGO computer program, their techniques were found to speed-up the evolution process and resulted in networks with better quality that was not limited to those of their opponents (Lubberts and Miikkulainen, 2001).

However, although the *number-of-generations* is the only theoretical limit to the quality in a co-evolution, a number of problems were still pointed out:

- For instance, stagnation can affect the co-evolution when new players get defeated by weak ones as the latter get filtered out of the evolving populations, and this thus leads to cycles by which the system cannot converge to an ideal solution (Rosin and Belew, 1997).

- Another prominent limitation to the application of neuro-evolution to some games is the computational cost. For instance, in the game of Go, the current practice is to evolve *9×9* boards at most, with most of the results reported on *5×5*.

- Finally, neuro-evolution may suffer from an evolutionary convergence to a local optima; widely known as *stagnation*. Stagnation is detected when evolution fails to improve on existing solutions as measured by the fitness function. In games, this in effect corresponds to a player that fails to improve its skills, and thus its performance remains constant.

Considering those concerns about the stagnation of co-evolution, Mayer and Maier (Mayer and Maier, 2005) co-evolved NNs of a *5×5* configuration, while they dynamically grew a group of the master players that appeared throughout the evolution. This group thus gathers and saves the knowledge *"culture"* of the population *(thus termed Cultural Co-evolution)*. The Elite – *related to the Hall-of-Fame concept (Lubberts and Miikkulainen, 2001)* – is a fixed group of players similar to those master players, also proposed as a co-evolutionary approach additional to the dynamically growing culture (Mayer and Maier, 2005). This method depended on a winning-rate, required up to *55,000* generations *(20 days of run-time)*, and was tested against a Random Player, a Naïve Player, and JAGO *(a Go program written in Java)*.

Evolving Go players without any prior knowledge about the game holds a promising foundation for scaling up to full-scale *'19×19'* Go (Richards et al., 1998).

For example, in (Stanley and Miikkulainen, 2004) the NN represented a scalable architecture, that used the *roving eyes*; each of which was a visual field that was more limited than the considered board but which could scan the board at will. The NNs were evolved on a *5×5* configuration against GNU Go using NEAT (Neuro-Evolution of Augmenting Topologies). The evolved roving eyes were then further evolved on a *7×7* board, building upon the experience gained from playing on the *5×5* boards. When compared to *7×7* boards that were evolved against GNU Go from scratch, the roving eyes did lead to a better performance than learning directly on the larger board.

Recent modifications to traditional neuro-evolution included a novel *Particle Swarm Optimization enhanced Evolutionary Algorithm (PSO-EA)* (Figure 2.12) and *Pareto Archived Evolution Strategies (PAES)*, both of which were employed to train the NNs. The PSO-EA hybrid training (Cai and Wunsch II, 2007) was proposed to achieve a better diversity along with a rapid convergence. The technique was applied to *9×9 Capture Go*. PAES (Tan et al., 2010) was proposed to account for two objectives while evolving the Go playing NNs; namely, 1) maximizing the playing capabilities of the evolved networks, while 2) minimizing the complexity – *in terms of the number of hidden-nodes* – of those networks. PAES improved the performance of the evolved NNs playing against GNUGO-3.6 on *5×5* Go boards.

Different board representations for a self-training ANN using *5×5* boards has been investigated by (Mayer, 2007). Besides the simple representation where each intersection is mapped directly to a single neuron, the author also tried capturing the neighbourhood information either by using the weighted sum of the values of the intersection and its *4* neighbours *(liberties)* or by using *3×3* overlapping squares positioned over the different intersections. On average, and using networks trained with temporal difference learning (Sutton and Barto, 1998), networks using the weighted sum of values representation outperformed those based on the other two representations.

Another alternative learning approach included combining Reinforcement Learning *(RL)* with Learning Vector Quantization *(LVQ)* (Abramson and Wechsler,

2003), which represents with its codebook vectors generalized and compressed input-patterns' templates. The concepts of RL and Patterns was also again used by Silver et al. (2007), where they based their RL approach on the linear evaluation of a large number of features based on local small-shape templates of the Go board, thus providing a quantitative estimate to those templates. By evaluating their program against computer opponents using *9*×*9* boards, they found that those translation-invariant small templates were effective.

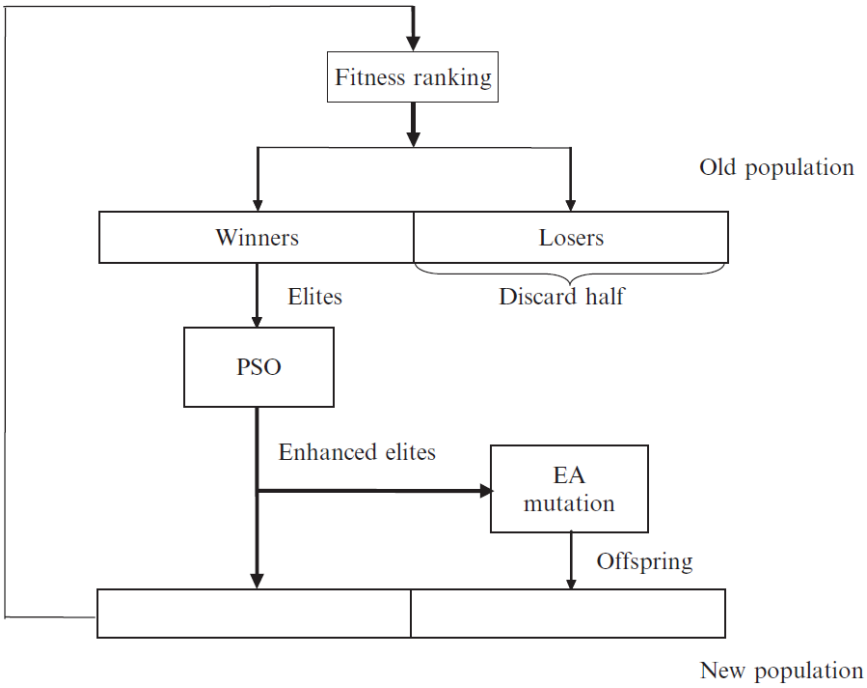


Figure 2.12. Flow chart of the hybrid PSO-EA method (After Cai and Wunsch II, 2007). The winners, which contain half of the population, are enhanced by PSO and kept in the population for the next generation. Those enhanced winners also work as the parents in EA to produce offspring. The offspring replace the discarded losers to keep a constant number of individuals in the population for the next generation. If the PSO block is removed, the hybrid algorithm is reduced to a conventional EA.

A three-component human-like-reasoning architecture for the Go Game was proposed by Meiger and Koppelaar (2001). It does not employ game-specific knowledge, and so their learning architecture HUGO "HUman GO" can be useful to

other two-player perfect-information deterministic combinatorial games. Its three components are a well-defined sub-games selector, an initiative engine that selects the best move based on both the potential initiative and the sub-game value which is calculated by the third component, a *fuzzy partial ordering based α-β* search algorithm.

A literature tree that provides a summary of the approaches to computer Go presented up to this point is shown in Figure 2.13. For more elaborate computer-Go surveys, the reader may refer to (Bouzy and Cazenave, 2001; Müller, 2001; Müller, 2002; Cai and Wunsch II, 2007; Chen et al., 2009).

2.2.5 Modifying the Game's Rules

In some instances, the rules of Go are changed to facilitate the proposed techniques for an automated computer player. In (Lubberts and Miikkulainen, 2001), it was assumed that all the stones remaining by the end of the game are alive, and hence that dead stones should be explicitly captured, thus eliminating the overhead of determining whether stones are alive or dead. In (Sei and Kawashima, 2000) the same assumption was adopted, but since *"according to the Japanese rules (Kiin and Kiin, 1989)"* a big eyeless group of stones is considered alive if after being captured a new living group of the same colour can be made in the resulting space, Sei and Kawashima adopted another method that a *"[c]omputer continues playing as long as there is a legal move"*. Playing until the end and statically considering a group of remaining stones alive if it cannot be proved dead is the so-called static recognition of life and death (Van der Werf et al., 2003).

Another vague rule is that of article 12 (Kiin and Kiin, 1989) that states that a game ends with *'no result'* when an identical whole-board position recurs, if the players agree to consider it a repetition. This vagueness is usually associated with the case of a *super-ko*; a longer cycle when compared to the immediate repetition of a cycle of only two moves *'basic-ko'*. To overcome the ambiguity arising from the undefined *'whole-board position'*, Sei and Kawashima (2000) describe it as when the *"following five items are all the same: arrangement of stones, player to move, position of Ko, whether*

the move immediately before is [a] pass or not and the difference in [the] number of prisoners." This description of the *'whole-board position'* can be called situational, when compared to positional ko-rules that detect repetitions based only on the arrangement of the stones (Van der Werf et al., 2003).

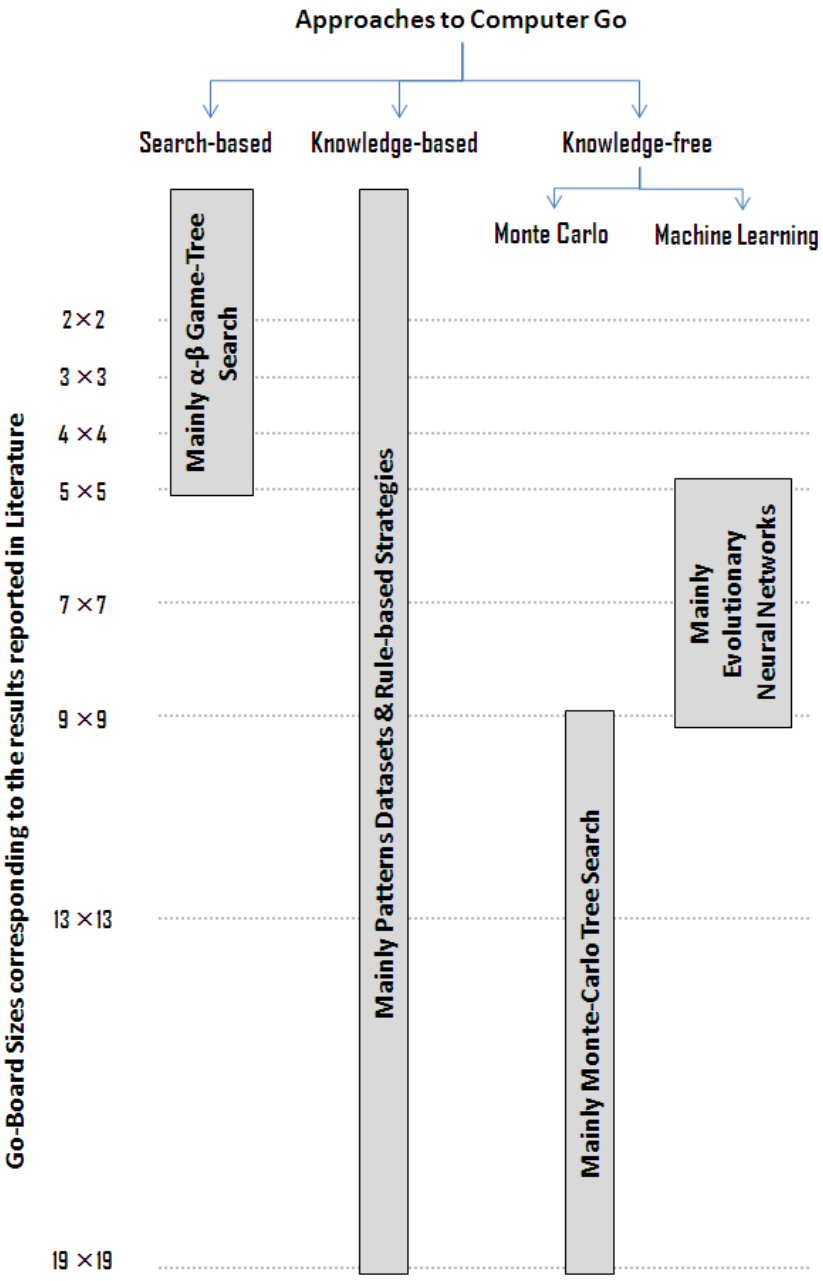


Figure 2.13. A brief Literature Summary Tree for the Approaches to Computer Go.

In (Lubberts and Miikkulainen, 2001), instead of storing all previous configurations for the temporal comparisons required by the *ko-rule*, the current state is compared to only two configurations ago, thus still grasping most of the *ko* situations with much fewer computations. In addition, they also placed an upper bound on the number of moves to forbid the case of an infinite loop.

Finally, according to the game rules, the end of the game is reached after successive passes of both opponents, and their agreement about the territory and *life-and-death* of the stones (Kiin and Kiin, 1989). The same scenario occurs with computer Go players, where if the programs disagree, the developers and then a referee are responsible for deciding upon this conflict. On the other hand, this overhead of agreement is pointless in search-based approaches when one computer program represents both players (Sei and Kawashima, 2000).

2.3 Summary

This chapter started by providing a basic introduction to the game of Go, followed by providing a timely review of computer Go and the different approaches utilized within, and the corresponding difficulties. Instead of the classical search-based approaches that have been successfully utilized to other games, yet failed within the Go domain, the approaches to computer Go can be categorized to knowledge-based approaches and knowledge-free approaches. The latter category includes random searches using Monte-Carlo based simulations which to date achieved the best results, and knowledge discovery using Machine Learning techniques. The main difficulties to those approaches include: the impracticality of encoding human expertise into the knowledge-based methods, the lack of adequate training procedures in addition to the high computational requirements for the Machine Learning methods, and finally, the difficulty of integrating domain knowledge into the Monte-Carlo based approaches to overcome the currently identified limitations.

The formalization and consequent implications of those difficulties will be investigated in the upcoming chapter (Section 3.1), including background knowledge on

the concept of Expertise. Following, is a review on the foundations of Competency, and its role in Situational Awareness and Management (Section 3.2). Subsequently, Situation Reasoning will be introduced in Section 3.3.

Chapter 3: Revolutionizing Competency Awareness

This chapter starts by re-examining the drawbacks of computer Go, providing an explanation of those drawbacks in terms of knowledge-acquisition and training bottlenecks. The explanation will take account of the nature of expertise as the basis of the existing challenges, and will conclude with the need to reconsider the concept of competency and the prevailing approaches to its measurement. A timely review of competency, the different approaches for its assessment, and its support to an overall situation awareness and management will be discussed in Section 3.2, which will conclude with an investigation of the disadvantages of the contemporary competency assessment methods, specifically in computer Go. To address those disadvantages, the necessity of a computational approach to competency assessment – *based on selected criteria; Patterns of Connectivity and Situation Reasoning* – will be established in Section 3.3. Consequently, Section 3.4 will provide a basic introduction to Patterns of Connectivity, followed by a basic introduction to Situation Reasoning in Section 3.5.

3.1 Revisiting the Challenges to Computer Go

> *No problem can be solved from the same level of consciousness that created it.*
>
> *–––Albert Einstein (1879-1955)*

As can be noted from the previous review on the approaches to computer Go, in spite of the recent achievements, computer Go still constitutes a challenging task. The challenges to the existing approaches can be summarized in three core points:

- Limitations of the Knowledge-based methods: these methods even though relying on operations which are – diversely – carried out by

human Go players (i.e., rule-based inference, generalization, pattern matching … etc.), have not adequately recognized and acquired those operations (Mańdziuk, 2008) or how they develop with experience (Harré et al., 2011a). In addition, the narrow scope and qualitative nature of the existing studies on Go expertise hinder further advancements towards surpassing human proficiency in computer Go (Harré et al., 2011b). For some of those operations, such as intuition, the available studies are still at a basic stance (Linhares, 2005). For those reasons, knowledge-based methods have virtually ceased from being an explicitly pursued approach in recent studies.

- Limitations of the Machine Learning methods: The view of Machine Learning approaches as an imitator of humans in their ability to learn from scratch without an explicitly incorporated knowledge-base has led to various attempts for developing "truly autonomous" computer Go players (Mańdziuk, 2007). Specially, in the light of the many successful applications of, artificial neural networks for instance, to many real-world applications, including board games (Tesauro, 1989). However, the nature and limits of the studies on human behaviour patterns and performance measurements have affected the implementation of an adequate training process, whether for explicit training as in supervised learning algorithms, or for evaluating and monitoring unguided development as in evolutionary algorithms.

- Limitations of the Monte-Carlo based methods: Driven by the disadvantages of utilizing both knowledge-based and Machine Learning approaches and their apparent limitations, present approaches in computer Go have mainly focused on Monte-Carlo based simulations, a process which is definitely not performed by human players. Although the research in this field has led to significant game-playing levels, a MC-based simulation by itself has fallen short of surpassing human Go

experts. Due to the unproductiveness of multiplying the number of simulations, up-to-date improvements have relied on augmenting MC-based approaches with experts' domain knowledge (Hoock et al., 2010; Rimmel et al., 2010).

3.1.1 A knowledge-Acquisition Bottleneck?

To further formalize those challenges, they can be expressed in terms of a *knowledge-acquisition bottleneck* (Murray, 1997; Wagner, 2006; Hoppenbrouwers and Lucas, 2009) and a *training bottleneck* (Hoffman, 1998; Ross, 2006). The knowledge-acquisition bottleneck (KAB) traditionally refers to the challenge of capturing and maintaining the required knowledge, and is usually discussed in the literature from an Organizational or Educational perspective. This bottleneck was described (Wagner, 2006) as: 1) the presence of constricted channels through which knowledge is acquired from its source, and which frequently lead to an 2) acquisition latency in which a delay exists between the creation and acquisition of knowledge, in addition to 3) inaccuracies that might result from experts and/or data-mining techniques, and finally, the 4) challenge of maintaining an acquired knowledge-base as it expands temporally. Obviously, though described from an organization-management perspective, the KAB characterizes several of the issues confronting computer Go.

The challenge of maintaining a domain knowledge-base – in its typical rule-based representation – is evident in virtually every knowledge-based Go approach. Besides issues such as the demanding number of patterns, and detailed tactical rules to be incorporated, the knowledge in those approaches is not well structured, and in frequent cases cannot be accurately explicated and formalized; as in the case of the Go-board evaluation which up to now requires innovative knowledge engineering approaches. Unsurprisingly, this is expected, given the fact that domain experts – traditionally employed for capturing domain knowledge – are themselves challenged by explicating their knowledge (Figure 3.1). This view was stated by Hoppenbrouwers and Lucas (2009); *"Although there are certainly many situations where knowledge discovery from data and machine learning can be very useful, the fact must be faced*

that learning technology will not resolve the KAB for cases in which highly domain specific knowledge (ultimately kept in individuals' minds) has to be made explicit and formalised."

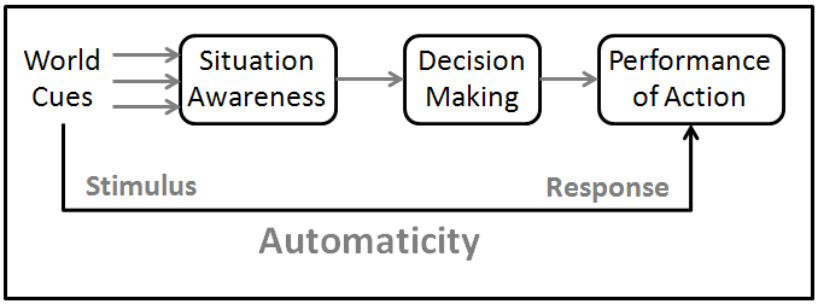


Figure 3.1. Automaticity in Cognitive Processes (After Endsley, 2000). Automaticity can be described from a cognitive view, as an occurring characteristic of expertise, in which – with experience – *"skills loses the quality of being conscious, effortful, deliberate, and linear, and takes on the quality of automatic pattern recognition"* (Hoffman, 1998).

Relying on experts also results in frequent inaccuracies and ambiguities, as is evident in recent Go studies (Hoock et al., 2010). This is due to the subjective qualitative nature of their understanding and assessments. The latter is caused by the dependence on think-aloud "verbal" procedures; which are pervasive for studying expertise in games (De Groot, 1946; Connors et al., 2011), and for the generic acquisition of knowledge (Wagner, 2006; Hoppenbrouwers and Lucas, 2009).

3.1.2 A Training Bottleneck?

However, and as we have previously mentioned in Section 1.2, a much more significant – yet relevant – challenge opposing the progress of computer Go is a *training bottleneck*. This concept is considerably discussed within the expertise literature (Klein and Hoffman, 1993; Hoffman, 1998; Ross, 2006), and its effect can be seen more on the Machine Learning based approaches to computer Go. The training bottleneck (TB) can be briefly defined as the challenge *"of getting knowledge into novices"* (Hoffman, 1998). In particular, it addresses the challenge of identifying the

limitations of an agent in performing a specific task with proficiency, and constructing corresponding procedures that would eliminate those limitations. Those procedures typically refer to training processes, but they can be simply extended to include knowledge that is explicitly added to knowledge-bases; ultimately, the aim is to augment an agent with a specific set of skills to overcome an identified drawback.

The relationship of the TB to the domain knowledge is twofold. Partially, the TB can be considered as a result of the KAB. The facts that some Go domain knowledge is yet to be discovered and acquired or that some available knowledge is yet to be properly explicated and formalized definitely increases the TB. Explicitly, by preventing – respectively – the identification and the acquisition of any missing skills and/or knowledge essential for the development of expertise.

Still, that is not the only reason for the TB; as there is still a substantial amount of Go knowledge available on many aspects of the game, from both current experts and historical games. Evidently, computer Go agents – specifically those based on Machine Learning approaches – are affected by the absence of suitable methodologies to perceive the missing set of knowledge, skills, and characteristics needed for the development of expertise. In other words, a major issue is the absence of competency assessments; methods that would identify and measure the competencies established within an agent against the set of competencies required for the proficient performance of a specific task.

3.1.3 How can Expertise be Organized and Measured?

So far, we have attempted to describe the challenges to computer Go from the training bottleneck point of view. Yet, a question remains, what are the reasons behind these issues? An answer is trifold, as it lies in the very nature of expertise, and consequently, in both the areas of the empirical studies on understanding expertise and the existing approaches to assess expertise. Expertise can be defined as what *"can describe skills, knowledge and abilities in [a task]"*, or, more from an AI perspective, it can be viewed as *"a structure [or] a system comprising knowledge and the rules that*

govern the relationships within a knowledge system" (Farrington-Darby and Wilson, 2006).

Several characteristics occur with expertise, and this thus can be used to characterise its existence and development. The development itself can be characterized by shifts in performance, and is theoretically accompanied by changes in knowledge organization – *for instance, how concepts are represented, interrelated, abstracted, or generalised* – and reasoning (Hoffman, 1998). The characteristics also include the advanced perception evidently exhibited by experts (Klein and Hoffman, 1993; Furse and Nicolson, 1993), which was one of the major conclusions from de Groot's (1946) study using Chess. An example of a generic perceptual capability which experts possess is in the *"ability to see typicality"*; that is, the ability to identify a typical situation from an exception, and consequently to recognize the significant cues, goals and expectancies, which ultimately lead to a swift and effective decision (Klein and Hoffman, 1993).

It is appropriate, in this context, to mention that experts can be categorized into epistemic and performative; respectively, those who are experts because of *"what they know"* and thus can provide strong justifications, and those who are experts because of *"what they do"* and thus can perform a skill with proficiency (Weinstein, 1993). Therefore, performative experts are not expected – and frequently do not have the capability – to describe how they perform a specific task with proficiency. This is analogous with an occurring characteristic of expertise, namely Automaticity (Figure 3.1) which was also termed a *"Declarative-to-Procedural Shift"* by Hoffman (1998), which can also hinder experts from describing and explicating their knowledge.

Nonetheless, studies about experience have frequently relied on verbal approaches; such as interviews, think-aloud protocols, or descriptions provided by direct observations. This is specifically evident in the many studies of expertise conducted using games, particularly Chess (de Groot, 1946; Newell and Simon, 1972; Gobet, 2006; Connors et al., 2011), which were based on qualitative and descriptive

results that were first gathered and then analysed statistically and/or interpretatively. Recently however, and using the game of Go, Harré et al. (2011a) introduced a technique based on information theory to better understand the progress of human expertise by measuring it quantitatively. The authors, utilizing data collected from skilled amateurs and professionals, quantified changes – developmental transitions – that occur as players' skill-levels progress.

In any case, measuring expertise can be categorized according to whether the foremost purpose is to identify the processes behind expertise or simply the outcome of those processes, or both. In the former case, the aim is to study, for instance, the processes of decision making or problem solving, while in the latter case, the studies are interested in performance-based clues to the dissimilarities between beginners and professionals by focusing of the outcomes, that is, the decisions and solutions per se (Farrington-Darby and Wilson, 2006). In view of performance measures, and considering the aforementioned portrayal of expertise by Farrington-Darby and Wilson (2006) – as *"what can describe skills, knowledge and abilities in [a task]"* – a competency assessment is an applied way to address this issue of *assessing the set of knowledge and/or skills required for performing a particular task with expertise*.

As a final point to this section, Figure 3.2 summarizes the concept of expertise from different studies in the literature; it shows the variations in the associated characteristics as expertise develops, and the dynamic factors behind this development. For a review of the literature on expertise, the reader may refer to (Farrington-Darby and Wilson, 2006). Meanwhile, to achieve our objectives, we need to review the prior research on competency, which will be presented in the upcoming section. Competency awareness cannot be fully understood without acknowledging its link to Situation Awareness and Situation Management, both of which will be subsequently reviewed.

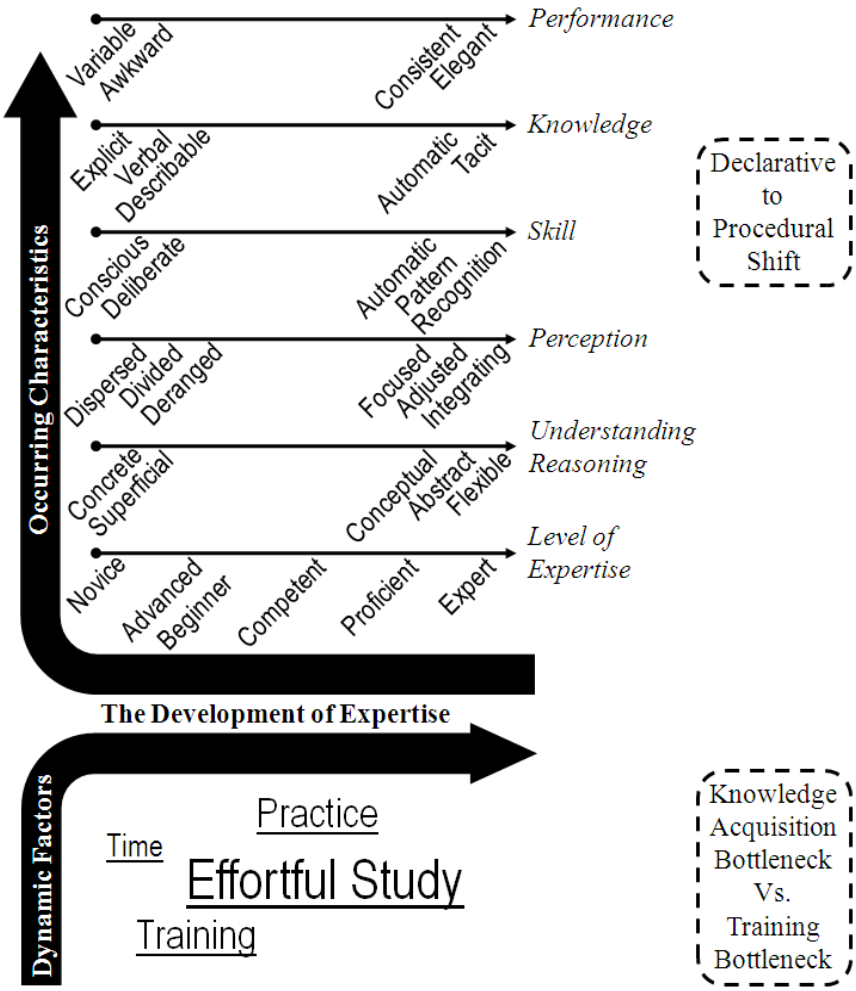


Figure 3.2. The Concept of Expertise.

3.2 Competency

> *To be conscious means not simply to be, but to be reported, known, to have awareness of one's being added to that being.*
>
> *---William James (1842-1910)*

Competency is typically investigated in the terms of evaluating and developing the human resources of an organization, perhaps this is because the bulk of the research on this topic comes from the Management and Organizational Development literature.

In addition, considerable research has come from the educational literature, where there is a point of view that stresses on performance outcomes and creating educational strategies to develop that outcome (Shippmann et al., 2000).

Therefore, many definitions for the term competency were formulated by this varying literature. Competency can be modestly defined as *"the set of behaviour patterns which are needed to allow the incumbent to perform tasks and functions with competence"* (Woodruffe, 1993). An overall competency is collectively established by component competencies, that is; skills, which refer to the learned capacities – whether general or domain-specific – that would be necessary or useful to carry out a particular job (Bassellier et al., 2001). Mirabile (1997) stated that competency is *"a knowledge, skill, ability, or characteristic associated with high performance on a job, such as problem solving, analytical thinking, or leadership"*, and that *"[s]ome definitions of a competency include motives, beliefs, and values."* Parry (1996) added that competency *"correlates with performance on [a] job"* and thus it *"can be measured against well-accepted standards, and that [it] can be improved via training and development."*

The variations in defining the concept of competency reflect different formulations of the issues that are intended to be addressed by specifying the competency. Those variations lead to a range of models that have been provided to integrate all of the competency's components. The foundation for competency modelling was laid by McClelland (1973); in which he criticized the use of traditional Intelligence tests as predictors of job success, and instead suggested analysing the actual components of proficiency in each particular job. Since then, a number of methods have been proposed for building competency models and assessing individual's capabilities, mostly in the human resources field to clarify the qualities – i.e., success factors – of effective personnel and managers (Table 3.1). For instance, Munro et al. (1997) evaluated the personnel's competences in using 'end-user computing' – such as specific software packages and tools – in terms of how many tools an individual was accustomed to, how deep was the individual's knowledge with respect to each specific tool, and how creative was he/she in utilizing the tool. More complete mechanisms to

measure the ‘end-user computing’ competencies were investigated in (Torkzadeh and Lee, 2003; Yoon, 2009), in which the authors, using surveys and factor analysis, proposed 12-item and 17-item scales – respectively – for measuring those competencies. Some studies focused on a specific occupation, system, or perspective; such as exploring information technology competencies in Business Managers (Bassellier et al., 2001), examining the role of the human resource system in managing and developing competencies (Lado and Wilson, 1994), or focusing on managers’ and employees’ view of each other’s competencies (Stoker and Van der Heijden, 2001). Additional studies were committed to examine the impact of external – yet relevant – factors on the measured competencies, such as the end-user involvement and the tasks’ uncertainties (Blili at al., 1998).

Traditionally, competency models – exemplified by Table 3.2 – are constructed using Interviews, Focus groups, and Questionnaires/Surveys (Mirabile, 1997), in addition to psychological assessments (Graham et al., 2003; Groth-Marnat, 2009) and experts feed-back (McClelland, 1973). Those constructing methods typically involve direct observations, video-camera recordings, and/or personal judgements/ratings (Mirabile, 1997). In addition, different layouts are available for constructing competency models (Mirabile, 1997), for instance:

- Models vary in their dependency on statistical data as opposed to behavioural descriptions.
- Models vary in whether success factors are identified independently or clustered in broad categories, and consequently, in whether the behavioural descriptions are provided – individually – per factor or listed – collectively – under the categories.
- Models also vary in whether or not proficiency-levels and/or criticality-ratings need to be established for each factor/cluster/description.

Table 3.1. A table relating a list of generic behavioural competencies to Roles (After Woodruffe, 1993).

Competency	**Output (Roles)**						
	Strategic Thinking	**Problem Solving**	**Persuasion**	**Staff Management**	**Training and Development**	**Customer Service**	**Business Development**
Breadth of Awareness; to be well informed	×	×		×			
Incisiveness; to have a clear understanding	×	×	×	×	×		
Reasoning; to find ways forward	×	×	×	×	×	×	×
Organization; to work productively		×		×	×	×	×
Drive; to achieve results		×	×	×	×	×	×
Self-Confidence; to lead the way	×	×	×	×	×	×	×
Sensitivity; to identify others' viewpoints		×	×	×	×	×	×
Cooperativeness; to work with other people		×	×	×	×	×	
Goal-Orientation; to win in the long term	×		×	×	×	×	

Though these models were designed to spell-out the requirements for a particular task, a significance of those methods is in identifying the limitations in knowledge and skills, thus ensuring a more effective targeting of those limitations, by designing tailored training experiences. The training resources may also be directed

towards developing the skills with a key influence or the skills which are more valuable during a particular phase of expertise.

Table 3.2. Part of *"a Competency Model for a Systems Engineer"* that identifies technical competencies and their corresponding potential proficiency levels (After Mirabile, 1997).

A Competency Model for a Systems Engineer [Technical Cluster]	
Technical Cluster	**Corresponding Proficiency Ratings**
1. System Architecture Ability to design complex software applications, establish protocols, and create prototypes.	0-Is not able to perform the basic tasks. 1-Understands basic principles; can perform tasks with assistance or direction. 2-Performs routine tasks with reliable results; works with minimal supervision. 3-Performs complex and multiple tasks; can coach or teach others. 4-Considered an expert in this task; can describe, teach, and lead others.
2. Data Migration	…
3. Documentation	…

Those benefits, even though reported from a largely managerial point of view, correlate with key requirements in the generic *training-experience* and *performance-measure* phases of Machine Learning algorithms. Modelling the human proficiency/behaviour directly affects many practical applications. Some recent applications vary from predicting navigation/driving behaviours and decisions (Ziebart et al., 2008), the behavioural cloning of human players to create interactive-video game-playing agents (Fabian, 2008), and to learning tasks demonstrated by experts (i.e., Apprenticeship Learning) in order to recover an unknown reward function (Abbeel and

Ng, 2004). The latter application is currently perhaps the only hope in the case of an extremely complicated reward function, a significant issue especially with *"outcome-evaluation"* as a major component in the naturally-pervasive value-based decision-making model (Rangel et al., 2008). For this latter application, an outcome-evaluation component can simply correspond to a performance-measure of the system's ability to learn the task demonstrated by experts, and thus the ability to choose the actions that maximizes—the possibly unknown—reward function. An improvement in the performance-measure will thereby reasonably indicate learning.

In conclusion, Competency models define the requirements to perform a task effectively in terms of a specific combination of skills, expertise, and knowledge (Lucia and Lepsinger, 1999). Consequently, a competency model then provides the awareness of an agent's capability of preforming a task, and the necessary tools for further training and development. Therefore, competency awareness serves the interests of decision makers by providing a needed factor for both genuine situation awareness, and accordingly more effective situation management. This role is underlined in the following section.

3.2.1 Situation Awareness and Management

Competency Awareness (CA) is thus a constituent of the overall Situation Awareness, which is an essential state for effectively accomplishing a task. The terminology Situation Awareness (SA) originated from aviation practitioners (i.e., aircraft piloting), where it is still a major objective (Durso and Sethumadhavan, 2008). In addition to other jobs which are highly cognitive – such as an air-traffic controller or an operator in power-plants – SA is used in constructing *"interfaces, automation control, and training systems"* (Endsley, 2000). Applications where SA is an objective are usually characterized by a flow of data that is changing swiftly. Amidst such environments, an information gap is present; that is, a gap between the genuine information required for decision making and the enormous collection of data which needs to be filtered, sorted and integrated within the dynamic functioning constraints (Endsley, 2000). Therefore, while designing automated systems, an objective is to

bridge this information gap by aiming for a more adequate SA and thereby assess to which degree that awareness is attained, and to also provide tools for maintaining that awareness.

Due to its dependency on the context, the definition of SA frequently varies due to the incorporation of specifics and terms of context. Endsley (1995) provided a generic definition of SA that stated its primary components; *"The [1] perception of the elements in the environment within the volume of time and space, the [2] comprehension of their meaning and the [3] projection of their status in the near future."* Thus, perceiving the relevant cues in the environment provides a basic level of SA, on which comprehending – integrating and interpreting – those cues conveys a more developed level, and finally, the capability of utilizing the current situation for anticipating future situations comprises the highest level of SA (Figure 3.3). Besides the primary components, there are factors that affect SA; these include time, limitations in the working memory, attention, and mental models, as well as the formulated goals and expectations (Endsley, 2000).

- Current state of the system (including all the relevant variables).
- Predicted state in the "near" future.
- Information and knowledge required in support of the crew's current activities.
- Activity Phase.
- Prioritized list of current goal(s).
    - Currently active goal, sub-goal, task.
    - Time.
- Information and knowledge needed to support anticipated "near" future contexts.

Figure 3.3. Elements of SA, given the situation (After Pew, 2000).

Owing to the necessity of attaining an awareness that is based on the present situation to develop effective decisions, SA modelling became an imperative component

of decision support systems (Feng et al., 2009). Modelling SA can be attempted from two perspectives; 1) Descriptive Models: in which the actual SA process is described either as a series of processes or in terms of situational and/or mental representations, and 2) Prescriptive Models: which are formal representations of processes with computational, numeric or simulation-based features (Banbury and Tremblay, 2004). As for assessing the SA, Pew (2000) provided a taxonomy of SA measurement methods; which include 1) Direct System performance Measures, 2) Direct Experimental Techniques such as probes, 3) Verbal Protocols and 4) Subjective Measures. The system performance measures directly relate a system performance to the SA attained, yet for this approach to be appropriately utilized domain experts should confirm that the system's performance is actually mostly determined by SA. The verbal protocols include think-aloud procedures and recording information, while the subjective measures include self-assessments, expert judgments, and ratings. Unsurprisingly, most of the SA assessment approaches can be effortlessly related to the previously mentioned methods for investigating expertise and for competency modelling/assessment. This can be attributed to the fact that attaining a proper SA is principally the result of expertise and training.

Awareness, whether of the overall situation or limited to competency and skills, can be fully understood only when presented within a general procedure by which a situation is managed, controlled, and decided. Recently, the existing concepts and operations associated with controlling a situation were analysed and combined – mostly from a cognitive perspective – into an explicated discipline; Situation Management (Jakobson et al., 2005). Situation Management (SM) was defined as the *"framework of concepts, models and enabling technologies for recognizing, reasoning about, affecting on, and predicting situations that are happening or might happen in dynamic systems during pre-defined operational time"* (Jakobson et al., 2007). The core theories of SM are based upon Modelling, Recognizing, and Reasoning-about situations (Jakobson et al., 2007). Nevertheless, SA must be taken into account (Figure 3.4) in order to provide effective decision making models; currently, for applications such as Disaster

Management (Chandana and Leung, 2010) and Battlefield Operations Management (Buford et al., 2010).

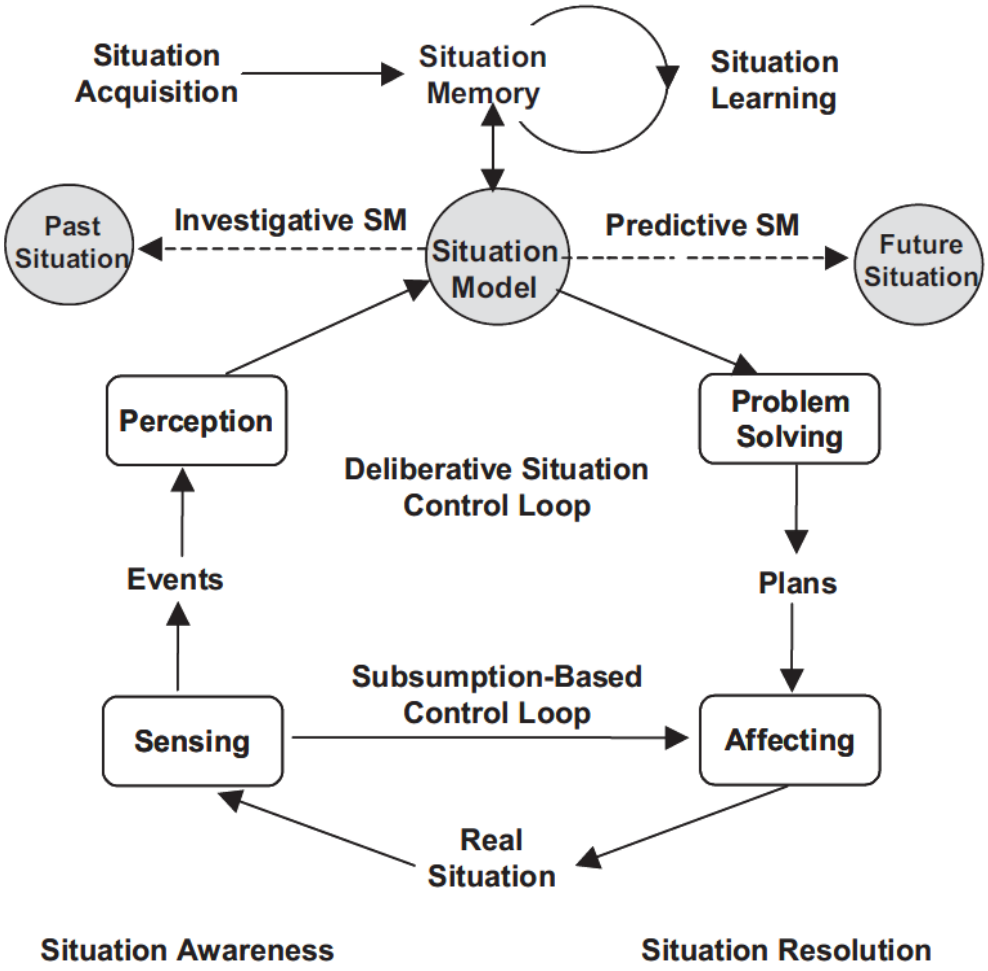


Figure 3.4. Situation Management – A General Process Loop (After Jakobson et al., 2007).

3.3 Awareness of Expertise in Go

Virtually all of the methodologies utilized for both Competency modelling and Situation Awareness are manual, relying as abovementioned on interviewing, questionnaires, direct observations, self-assessments, probes, verbal 'think-aloud' protocols, peer ratings, and expert judgements. The disadvantages of such methodologies are noticeable; lengthy data-collection processes, time-consuming,

subjective nature, disruptive effect, in addition to several other detailed drawbacks (Mirabile, 1997; Pew, 2000). It is reasonable to assume that these disadvantages also appear in studies of expertise that rely on no different methodologies.

The subjective nature of the Go knowledge that is explicated by human experts is evident in the recent computer Go research (Hoock et al., 2010). As for the disruptive effects of what can be called an "Active" or a "Direct" methodology (Figure 3.5), Pew (2000) for instance stated – in regards to SA – that these methodologies might *"[place] the subject in an unrealistic setting and producing an unrealistic assessment"* and/or *"[change] the subjects' behaviour."* A passive methodology, on the contrary, is where the observer is situated outside the system and thus does not influence – for instance – the interaction between a subject and its environment.

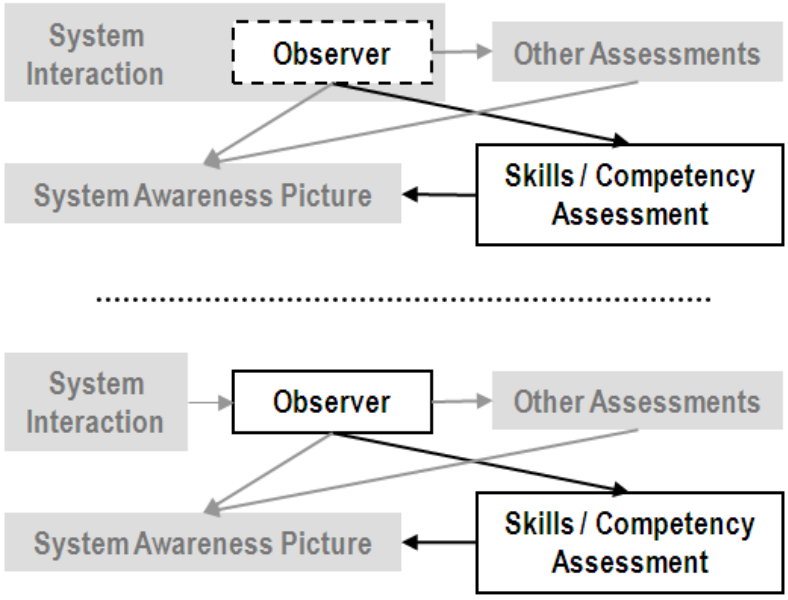


Figure 3.5. The Difference between a Passive (Top) and an Active (Bottom) Methodology. Modules that are out of the proposed framework's scope are in light grey. This notion of passiveness will be used throughout the thesis.

Therefore, the need is critical for an automated, non-disruptive/passive/non-intrusive methodology that objectively and quantitatively assesses the diverse skills and characteristics of the subjects' decisions. Such a methodology can also address similar

requirements for any advancement in AI; that is, providing significant quantitatively measured information on human expertise and decision-making (Harré et al., 2011b).

Unfortunately, in artificial systems, assessing the competency of an agent usually depends entirely on the degree to which the goals are achieved; that is, the final outcome. As in the domain of board-games, where ranking systems – *whether online or offline* – are practically the only objective approaches to evaluate the expertise of players. Consequently, existing automated methods lack many of the advantages of the manual ones; such as, obtaining a practical comprehensive picture, providing in-depth/detailed data, and capturing infrequent behaviours (Mirabile, 1997). Furthermore, in reality relying on a final outcome is – in many occasions – not possible and also incorrect. Let us take as an example the defence strategic planning domain. For it, the timeframe for outcomes to materialize is beyond the decision maker's lifetime. In addition, the quality of a decision depends on the information and circumstances surrounding the decision at the time it is made, not necessarily at the time outcomes become known.

Nevertheless, an advantage of the ranking systems in games – *the ELO System in Chess and the Kyu-Dan system for Go* – is that they provide a quantitative definite distinction in skill-levels, thus setting performance standards and deciding how an agent is ranked. This advantage is particularly convenient to studies of expertise that utilize those games. It is worth mentioning that this type of ranking – used within games – is analogues to the "Paired-comparison ratings" used for ranking competencies and employees within a job (Mirabile, 1997).

So, we hypothesize that we cannot anymore just rely on human experts or on the existing automated rating systems. The first step in order to realize and develop the desired competency assessment methodology – *which is entirely automated and passive, and provides objective quantitative in-depth information that has a potential for characterizing an agent's expertise and of differentiating between novices and professionals* – is to select skill(s)/competence(s) that can be investigated and evaluated.

The chosen competence(s) will then act as a typical outline/profile against which an agent's competency can be rated. In reality, and as in the Go domain, skills can exceptionally vary from noticeable to much less tangible and from simple to considerably complex.

A possible capability that can be investigated is pattern-connectivity. In contrast with systematic analysis and declarative reasoning, experts develop – *with experience* – a more tacit and automatic type of structured knowledge (Klein and Hoffman, 1993; Ross, 2006) that can be examined by exploring the present forms of repetitive objects or events. In the game of Go, the role of patterns is more obvious, where due to the nature of the game, it requires an intensive reliance on patterns to efficiently recognize, evaluate, and decide upon specific situations (Bouzy and Cazenave, 2001; Müller, 2002).

An alternative perspective is that of investigating *Reasoning*. Reasoning strategies and processes are clearly a major and well-established aspect of defining and judging expertise (Klein and Hoffman, 1993; Hoffman, 1998). Analysing the reasoning of Go players is firmly in line with de Groot's (1946) core investigational methodology for Chess; *"to ask subjects to think aloud when thinking about their next move"* (Gobet, 2006). In consequence, de Groot's proposed *'Chess Player's Thinking Process'* emphasized the role of reasoning, specifically in the latter stages where a more in-depth *investigation* of significant potential moves is carried out, followed by a *proof* stage that reviews and validates the analysis and arguments behind a selected course of action (Gobet, 2006).

In conclusion, to finalize the conceptual background for this study, Connectivity-Patterns will be discussed in more details in the upcoming Section 3.4, followed by Situation Reasoning in Section 3.5. We will then discuss specifically Connectivity-Patterns within Go in addition to the proposed approach within Chapter 4. Situation Reasoning for Go, as a variation to the proposed approach, will be discussed within Chapter 5.

3.4 Patterns of Connectivity

> *For there are two modes of acquiring knowledge, namely, by reasoning and experience. Reasoning draws a conclusion, but does not make the conclusion certain, unless the mind discovers it by the path of experience.*
>
> *–––Roger Bacon, Opus Majus (1268)*

Patterns, a terminology used within varying contexts, can be generically defined as being *"a compact and rich in semantics representation of raw data"* (Catania et al., 2005), or as *"an abstraction from a concrete form which keeps recurring in specific non-arbitrary contexts"* (Kumar, 2003). Patterns can thus act as templates or models to describe, predict, and/or generate this form – *in terms of the constituent entities, relationships, and/or attributes* – within the specific domain, mostly to address a problem within that – *or a relevant* – domain.

Patterns characterize a main aspect of human expertise, by numerous psychologists, in which experienced decision-making is based upon an extensively developed collection of *"patterns and actions conditioned upon them"* (Harré et al., 2011a). Within the wide existing spectrum of research on Chess, patterns within the game were investigated as a part of studying the players' decision-making, perceptual, and memory processes (de Groot, 1946; Gobet, 2006; Ross, 2006; Connors et al., 2011). Human's sophisticated skills of manipulating patterns have inspired – *and still lead* – the development of machines that can reliably perform these complex tasks of pattern recognition and classification (Duda et al., 2001).

However, beside those relatively well established fields *(such as Pattern Recognition)*, current studies include innovative approaches for analysing patterns. For instance, a recent study within the Go domain applied techniques of information theory to a selection of Go patterns in order to investigate the uncertainties that a Go player has when playing, and how those uncertainties develop with experience (Harré et al.,

2011a). Currently, a more generic approach to investigate patterns is to recognize patterns of connectivity, and the nature of their occurrences. That is, within a network that represents a specific domain – *or the dynamics of that domain* – by connecting the related entities, to search for small local patterns of connection, and to investigate the representation of those patterns within that network. Identifying those local patterns of connectivity – *termed Network Motifs* – have demonstrated usefulness in understanding the dynamics of the system that corresponds to the network being analysed (Ghoneim et al., 2008; El-Fiqi et al., 2011; Beck et al., 2011). In Section 5.1.1, more details on pattern identification within the Go domain – *in addition to network motifs* – are presented.

3.5 Situation Reasoning

Reasoning, within a specific environment, can be defined as the ability to perceive and comprehend the environment and its components, to find, create and validate its facts, and to – consequently – rationalize and/or alter beliefs and actions. Hence, reasoning can refer to the thinking and rationalizing process itself, or to a – corresponding – output of that process; such as a proof, an explanation or an argument. In this context, the interest is in the latter perspective; where in dictionary form, a reasoning/reason is *"a consideration which explains or justifies some event, phenomenon or behaviour"*.

As can be noted from the definition, and typically from a philosophical perspective, reasoning within that context is often classified as the reasons that justify why an event has to occur or what counts in favour of considering that an agent has to perform a particular action *(and hence, they are called justifying or Normative Reasons)*, and reasons that explains why an event has occurred or what motivated an agent to perform a particular action *(that is, Explanatory or Motivating Reasons)* (Tilley, 2004). Obviously, we are interested in the latter view, which is echoed in the literature through other relevant terminologies; such as a cause, causal history or

knowledge (Kim, 1981; Lewis, 1986; Garcia-Retamero and Hoffrage, 2006), or an explanatory argumentation (Amgoud and Prade, 2005).

As opposed to a cause, causal history is the set of causes that in association would lead to an unavoidable or – *at least* – a very much expected occurrence (Lewis, 1986). An example of a causal model for effective project management can be found in (Loo, 2003); the model designates specific practices – *in terms of technical and personnel skills* – as causes for some desired organizational outputs. Nonetheless, it is noteworthy that there are opinions against considering "causes" as a synonym for "explanatory reasoning", or at least the very simple causes. The presence, in a complex system, of *"many intermediate steps that intervene between cause and effect, which may not be linear"*, in addition to potential emergent properties, renders impractical any simple causal explanation of that system (Hmelo-Silver and Pfeffer, 2004). For that reason, it was argued that the concept of causality does not probably apply to nonlinear systems (Wagner, 1999). A critique of using causes as explanations can be found in (Kim, 1981). It is worth mentioning though that, in this work we are interested only in the immediate explanatory reason(s) of a particular action, not the chain of reasons – *behind the sequence of decisions* – that leads to the action under consideration.

As for reasoning as an explanatory "argumentation", though the terms are from time to time used interchangeably, dissimilarities between both terms have been widely discussed. From a philosophical perspective, reasoning consists of the foundation or evidence(s) in support of an argument which – *on the other hand* – is a tool of persuasion (Walton, 1990). In other words, as described in (Govier, 1989): *"Reasoning is what you may do before you argue, and your argument expresses some of your (best) reasoning. But much of the reasoning is done before and outside the context of argument."*

3.6 Summary

To sum up, in this chapter, the challenges to computer Go were revisited, resulting in a characterization of those challenges in terms of Knowledge-Acquisition

and Training bottlenecks, and which led to an investigation of Expertise and how it can be identified and measured. From the investigation, the merits of reconsidering competency assessments were covered, and consequently, competency, how it is modelled and assessed, and its role in the overall awareness and management of a particular situation were reviewed. From which, it has been found that current methodologies for competency assessments have limited capabilities that need to be addressed for further advancements in computer Go. Finally, connectivity-patterns and situation reasoning were selected as potential competencies to be investigated and evaluated. Connectivity-Patterns within computer Go, an explicated problem statement, and the specifics of the proposed methodology will be presented in the next Chapter.

[This Page is Intentionally Left Blank]

Chapter 4: Competency of Human Go Players in terms of Connectivity-Patterns

> *To know means to record in one's memory; but to understand means to blend with the thing and to assimilate it oneself.*
> *–––Ancient Egypt – Luxor/Karnak Temples Complex; de Lubicz, Her-Bak: The Living Face of Ancient Egypt (1978)*

In this chapter, a novel approach is introduced for quantifying and monitoring the connectivity-patterns employed by the human Go players. The chapter starts by an overview of the problem to be tackled and its formal definition. In Section 4.2, the methodology proposed to attempt this problem is presented, including an overall framework, and a range of connectivity-patterns (i.e., network motifs) to be assessed. Section 4.3 investigates the potential of the assessed motifs to capture and reflect the competency of Go players using a selected training set and simple statistical methods. The statistical methods include measures of dispersion to characterize the changes in motifs-counts among a selection of varying performance-levels (i.e., rankings), followed by tests of significance on whether the calculated measures can reliably distinguish those between performance-levels. Once this discriminating capacity of the assessed motifs is established, the calculated measures are used as features to construct a proposed classifier (Section 4.4). The classifier is then used to monitor the competency of selected human Go players – a testing dataset – as their expertise develops with time. Section 4.4 also includes a discussion of the achieved results. The contributions in this chapter are summarized in Section 4.5.

4.1 Overview

As has been determined in the previous chapter, the objective is to present a computational framework that is capable of automatically assessing the competency of

Go players; *which will attempt to answer the research sub-questions 1 and 2 (see Section 1.3)*. The automation of this task should reduce the need for invasive, subjective, and qualitative assessments by human experts. However, though automated, the framework is expected to provide relatively a more detailed assessment when compared to the existing automated approaches. In this chapter, the competency of Go players is to be assessed in terms of connectivity-patterns.

4.1.1 Related Work

Within the Go domain, patterns are widely utilized – *whether augmenting other approaches, such as Monte-Carlo simulations, or less successfully on their own* – to suggest a course of action, and as a way to overcome the complexity of formalizing a board-evaluation function (Müller, 2001; Abramson and Wechsler, 2003). Those connectivity-patterns exist in terms of datasets that are categorized according to specific functionalities; that is, Fuseki patterns, End-of-the-Game patterns … etc. Due to the nature of the game, each pattern exists in an explicitly-detailed manner, and is usually applied to a precise board-situation. However, and to the best of our knowledge, those patterns have not been employed in characterizing the performance of Go players.

A remarkable alternative exists within the generic literature, in that many complex systems are represented via networks – *"collections of points joined together in pairs by lines" (Newmann, 2010)* – that are then more deeply analysed and understood through a *Network Motifs* count. Network motifs can be defined as *"small recurring interconnections (subgraphs of three to five nodes) that appear more significantly in real networks than in randomly generated networks having the same single-node characteristics"* (Ghoneim et al., 2008). Networks motifs have been successfully employed in understanding the dynamics of numerous varying complex-systems; for instance: the dynamics within computational linguistics (El-Fiqi et al., 2011), the dynamics of gene sequencing and regulation (Beck et al., 2011), and the characterization of the dynamics of two-player strategy games in game theory, such as the prisoner dilemma (Ghoneim et al., 2008). This understanding is based on the most significant consequences of a network motifs count, which are 1) the assumption that

those counted motifs are developed as a result of similar factors and consequently perform similar tasks within the complex system, and hence, 2) the capability of clustering the analysed networks into groups of related characteristics (Ghoneim, 2008). Section 5.2 provides details on the main types of connectivity-patterns, followed by our proposed reasoning assessment approach.

4.1.2 Problem Statement

The problem to be attempted in this chapter is: to automatically, passively, quantitatively, and objectively assess the competency of Go players by characterizing the decision-making performance of human Go players in terms of connectivity-patterns that represent existing board situations. The connectivity-patterns are determined using a network motif search; that is, given a board situation, each individual-player's stones are represented by a network of nodes *(representing the stones)* and edges *(representing the adjacency between the stones)*, which is then searched for sub-graphs of specific sizes and patterns. This procedure is performed using a fully computerized simulation, and is used to analyse a large cohort of games played by – Casual, Intermediate, and Advanced – human Go players. The results expected to be achieved are:

- That given a set of Go *"Games"* performed by human players of varying *"Expertise"* levels, the difference in the expertise can be explained by a statistically-significant quantitatively-based difference in the frequencies of connectivity-patterns *"$Motifs_{Expertise}$"* estimated from the corresponding *"$Games_{Expertise}$"* using the network motif search.

- And consequently, that the quantitative measurements representing a specific *"Expertise"* level can be utilized to create a corresponding *"Competency Model"*. So that, given a set of Go *"Games"* played by a particular *"Player"* over a period of time, the *"Competency Model"* will reflect and monitor the changes in the performance of that specific *"Player"*.

4.2 Methodology

To accomplish this, we present the computational framework in Figure 4.1. The cornerstone of this modified framework is a pattern assessment component (AC_P), which will act as the *external observer*, and consequently, provide an assessment – *in terms of network motifs* – to the actions of the Go players. The AC_P depends on a systematic search for specific sub-graphs within typical networks. Therefore, this component can act as an objective *reference point* by which the pattern-connectivity competency of different players can be compared. The fact that the AC_P is fully automated ensures the objectiveness of the provided analysis. It also ensures the passivity and non-intrusive nature of the analysis, guaranteeing that it – that is, the analysis process – could not have an effect on the dynamics of the game being assessed.

With the purpose of examining the effectiveness of the searched motifs in expressing the competency of Go players, and therefore in discriminating between the varying levels of performance, our methodology includes a preliminary phase that attempts to characterise those varying levels in terms of the network-motifs totalled. This phase – steps from 1 to 3 in Figure 4.1 – includes the selection of Go games' records reflecting the different available levels of performance to constitute a *training dataset*. Those games are then analysed using the AC_P, and simple statistical dispersion measures – *the Median, and the Median Absolute Deviation* – are then calculated, and are followed by tests of Statistical Significance. The purpose of applying the significance tests is to demonstrate the value of the motifs as features capable of describing and – accordingly – distinguishing the varying levels of performance in Go.

Once this phase is realized, the features are then used to construct a classifier capable of learning the different recurring connectivity-patterns. The design of the classifier (Figure 4.5) is founded on the concept of Ensemble Learning (Kittler et al., 1998), and is constructed using Random Forests (Breiman, 2001). The construction of the classifier begins the second phase of our proposed methodology – steps from 4 to 6 in Figure 4.1 – which also includes the selection of a *testing dataset*; a selection of cases – *human Go players* – that will be used to monitor the competency as the players'

experiences vary with time. Temporally-ordered games are compiled for each case, and are analysed by the AC_P to generate the set of features to be classified. Finally, the votes from the trained classifier – *a vote for each game* – are plotted to show how the learning process is evolving, and to estimate the current overall competency/experience level for each selected player. A temporal plot of the features can present a more detailed skill assessment for the selected players. The remaining sections in the chapter describe – *consecutively, as shown in Figure 4.1* – each step in the framework. Coming next are the definitions of the classes of motifs available in the game of Go.

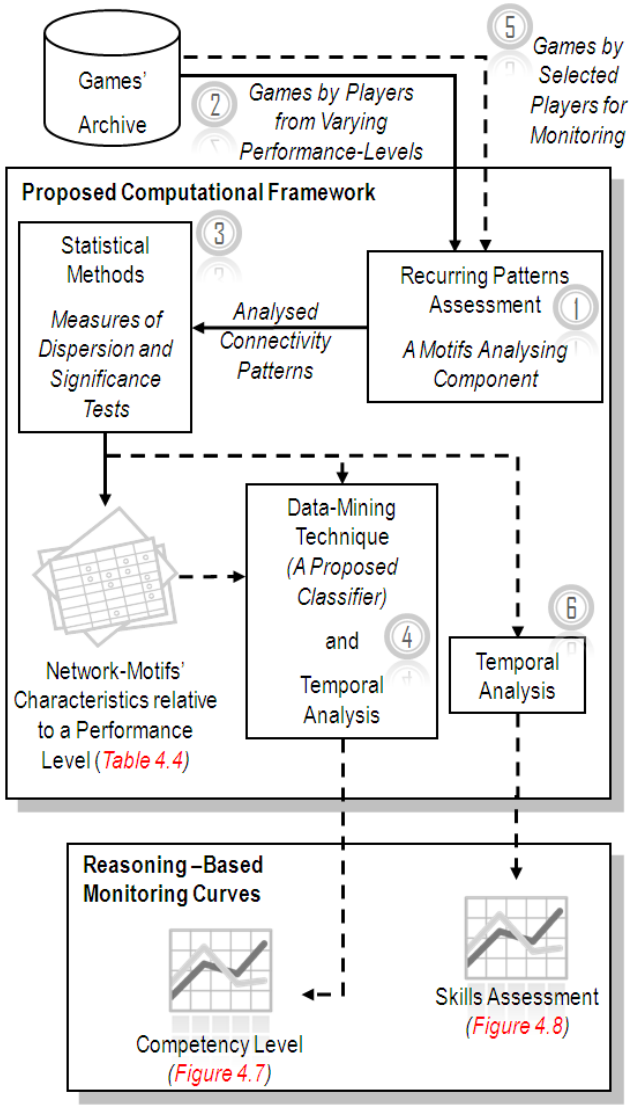


Figure 4.1. The Proposed Computational Framework.

4.2.1 Classes of Motifs

As previously defined (Section 3.4), motifs are small connected sub-graphs – that is, sub-networks within a studied network – that are displayed in significantly higher frequencies within that network than would be expected for other random networks. Therefore, varying classes of motifs can be found, depending on the criteria of the network and sub-graphs; namely, the type of connectivity that exists between the nodes in a network and the size of the sub-networks/sub-graphs we are looking for.

Table 4.1. Total number of Sub-graphs available.

	Nodes *n* per Sub-Graph		
	3	**4**	**5**
Total number of connected Directed-graphs with *n* nodes	13	199	9364
Total number of connected Undirected-graphs with *n* nodes	2	6	21
Number of connected Undirected-graphs with *n* nodes available in *trainDS*	2	6	15

Table 4.1 lists the total number of sub-graphs available, when searching for sub-graphs of sizes 3, 4, and 5 nodes '*n*', and also when searching for directed sub-graphs in which each edge has a sense of direction from a node n_{source} to $n_{destination}$ *(usually expressed as an ordered-pair* $< n_{source}, n_{destination} >$*)*, as opposed to undirected sub-graphs in which the sense of direction does not exists *(and thus, usually expressed as an unordered pair* $\{n_{source}, n_{destination}\}$*).* The networks representing a game of Go are built for every step – *move* – in the game, and per opponent; that is, a game of Go with *300* moves will results in *600* networks, thus for each action and after removing the dead/captured stones, the board is represented in *2* networks that express the White stones and the Black stones, individually. Hence, the stones of the same colour in a specific board situation constitute the nodes within the constructed network, while the

edges express the connectivity between those stones. In this context, connectivity is defined as stones – *of the same colour* – which are horizontally, vertically or diagonally adjacent.

In this study, undirected sub-graphs of sizes *3*, *4*, and *5* will be investigated. In a preliminary experiment, and using the training-dataset *'trainDS'* (described in Subsection 4.3.1), not all of the theoretically possible combinations of sub-graphs from the corresponding sizes (see Figure 4.2) were present in the networks representing the Go games. The last row in Table 4.1 shows the actual number of undirected sub-graphs found in *trainDS*. As a final point, the initial sets of sub-graphs to be investigated are:

- *Undirected Sub-Graphs of Size 3*: { M3-1, M3-2 }.
- *Undirected Sub-Graphs of Size 4*: { M4-1, M4-2, … M4-6 }.
- *Undirected Sub-Graphs of Size 5*: { M5-1, M5-2, … M5-15 }.

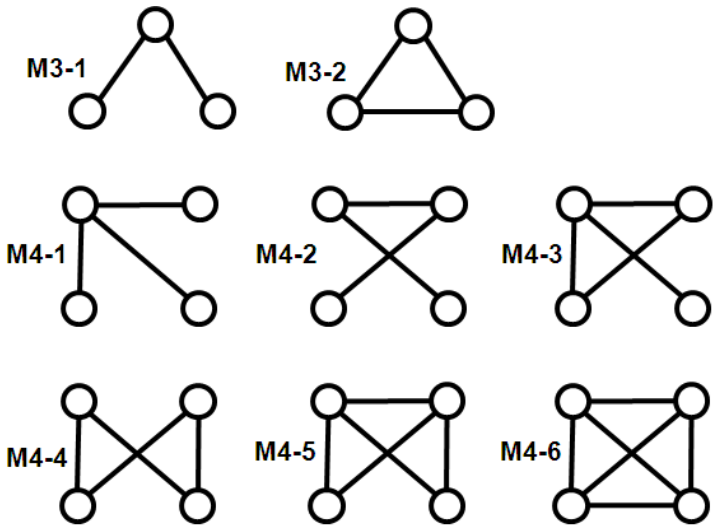


Figure 4.2 (a). Possible combinations of undirected sub-graphs of sizes *3* and *4* respectively, and their corresponding designated IDs used throughout the chapter.

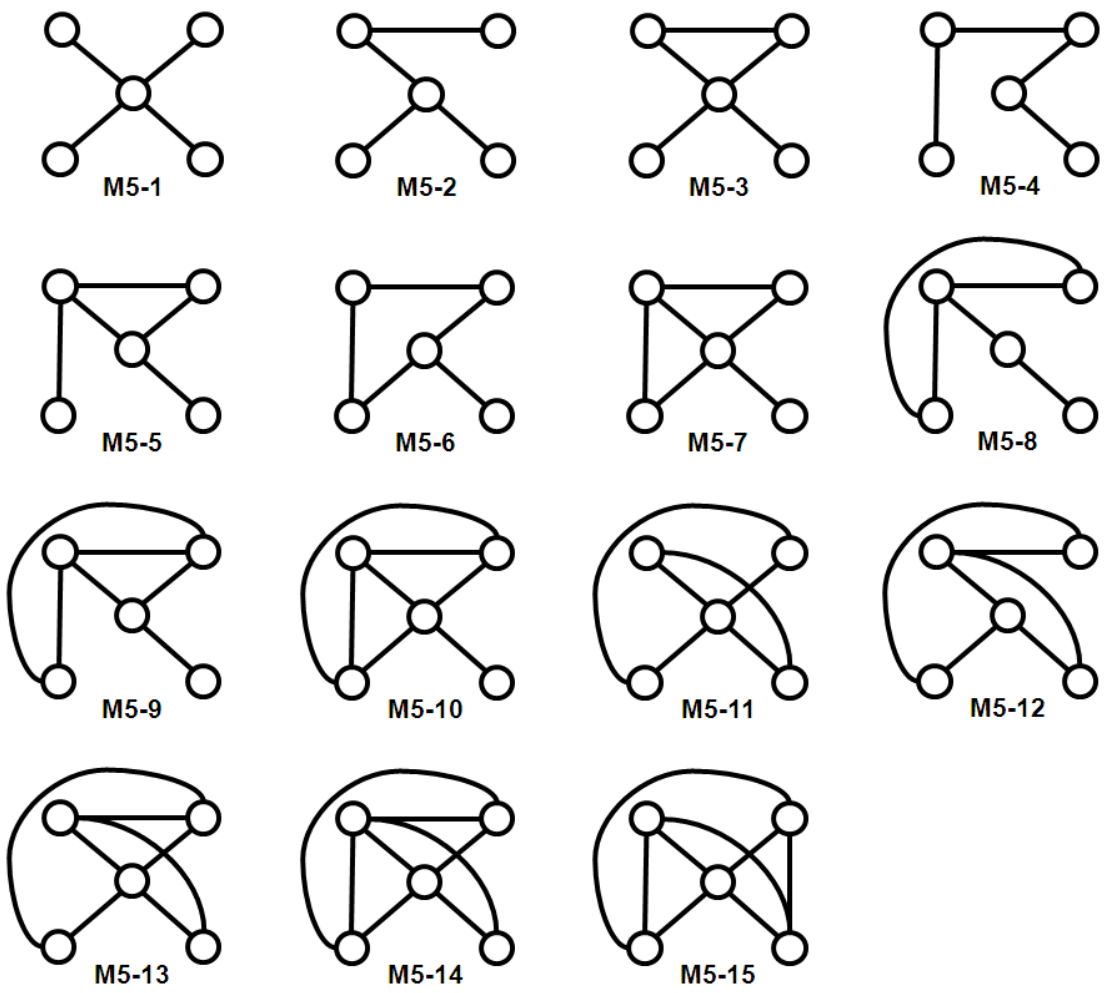


Figure 4.2 (b). Possible combinations of undirected sub-graphs of size *5*, and their corresponding designated IDs used throughout the chapter.

4.2.2 Nature of the Network-Motifs occurring in Go

As mentioned earlier, for a sub-graph to be considered a Motif, its occurrence in the investigated network must be significantly higher than its frequency in randomized networks. Thus, allowing the assumption that those significantly recurring patterns perform necessary tasks within the domain reflected by that network. In order to extract what the motifs are, from a Go perspective, the frequencies found in trainDS should be statistically compared to those found in proportional randomized networks. By proportional networks, we mean randomized networks that share the same

characteristics as the real network; such as the number of nodes, and the degrees of those nodes.

To do this, for each real network representing the stones of an individual opponent at a single step of a Go game, *500* proportional randomized networks were created. Then, the averages – *and the corresponding standard-deviations* – of the occurrences for each motif in all the random networks are calculated. In both Tables 4.2 and 4.3, the numbers of occurrences in the real *'Go'* networks are termed *'N real'*, while those of the randomized networks are termed *'N rand'*, and their corresponding standard deviations are termed as *'SD'*. Hence, statistical significance is represented in terms of the *Z-Significance* value, which can be calculated using the following equation (Ghoneim et al., 2008):

$$\text{Z-Significance} = (\text{N real} - \text{N rand}) / \text{SD} \qquad \text{(Eq. 4.1)}$$

This process is usually done by publicly-available dedicated software, such as FANMOD (Wernicke and Rasche, 2006). The specified cut-off threshold for the *Z significance* value is set to *±1.96*; that is, differences in the occurrences of a specific sub-graph – *between the real and randomized networks* – that yield a value of significance that are higher or lower than the selected threshold, are considered sufficiently significant for the corresponding sub-graph to be considered a Motif. This threshold corresponds to a confidence level of *95%* (El-Fiqi et al., 2011).

Table 4.2 presents the details – *of this Motifs identification process* – for *3* games, each representing a level of expertise. On the other hand, Table 4.3 presents the final overall results of the Motifs identification process for all of the games included in *trainDS*; therefore, the values shown in the table are averages of *127* games per expertise-level. Using the calculated values of significance, and the selected threshold, *3* sub-graphs – *1* of size *4*, and *2* of the size *5* – were not found to be Motifs *(shown in the Table in light grey cells).*

Table 4.2. Motifs Identification Process; *Details of 3 Representative Games*.

Sub-Graph Size	Expertise-level of the Representative Game	Motif ID	N real	N rand ± SD	Z Significance	Significant
3	**Casual**	1	251	420.5±4.5	-37.79	YES
		2	59	2.5±1.5	37.79	YES
	Intermediate	1	330	565.2±5.7	-41.55	YES
		2	82	3.6±1.9	41.55	YES
	Advanced	1	353	641.2±5.9	-49.26	YES
		2	100	3.9±2.0	49.26	YES
4	**Casual**	1	51	268.4±9.1	-23.96	YES
		2	366	1010.6±26.1	-24.69	YES
		3	141	15.4±8.9	14.13	YES
		4	8	4.8±2.1	1.52	NO
		5	40	0.1±0.3	122.46	YES
		6	3	0.0±0.0	INF.	YES
	Intermediate	1	64	410.4±13.3	-26.07	YES
		2	486	1523.0±34.7	-29.90	YES
		3	215	24.2±13.0	14.69	YES
		4	13	7.7±2.7	1.95	NO
		5	68	0.2±0.4	152.23	YES
		6	5	0.0±0.0	INF.	YES
	Advanced	1	71	494.4±13.6	-31.05	YES
		2	523	1774.4±38.2	-32.78	YES
		3	251	27.2±13.3	16.87	YES
		4	9	8.5±2.8	0.19	NO
		5	84	0.2±0.5	176.41	YES
		6	8	0.0±0.0	INF.	YES
5	**Casual**	1	4	98.2±7.9	-12.00	YES
		2	218	1905.3±111.7	-15.10	YES
		3	14	10.4±7.5	0.48	NO
		4	525	2390.6±110.0	-16.96	YES
		5	98	27.1±19.9	3.56	YES
		6	39	36.5±17.2	0.14	NO
		7	34	0.3±0.9	35.70	YES
		8	204	33.7±22.0	7.74	YES
		9	88	0.3±1.2	74.98	YES
		10	4	0.0±0.0	INF.	YES
		11	19	0.1±0.3	57.13	YES
		12	6	0.5±0.8	6.64	YES
		13	15	0.0±0.0	335.37	YES
		14	8	0.0±0.0	INF.	YES
		15	3	0.0±0.0	INF.	YES
	Intermediate	1	2	161.4±11.7	-13.68	YES
		2	226	3236.4±158.2	-19.03	YES
		3	20	20.5±10.8	-0.05	NO
		4	706	4012.2±152.6	-21.67	YES
		5	143	55.5±30.2	2.89	YES
		6	49	65.8±23.8	-0.71	NO
		7	60	0.8±1.8	32.29	YES
		8	321	68.6±34.2	7.39	YES
		9	142	1.1±2.5	57.43	YES
		10	5	0.0±0.0	INF.	YES
		11	20	0.2±0.5	37.14	YES
		12	20	1.1±1.3	14.96	YES
		13	39	0.0±0.1	617.20	YES
		14	16	0.0±0.0	INF.	YES
		15	5	0.0±0.0	INF.	YES
	Advanced	1	6	222.0±13.9	-15.52	YES
		2	272	4003.4±171.1	-21.80	YES
		3	31	25.7±13.0	0.41	NO
		4	689	4793.3±163.5	-25.11	YES
		5	166	64.0±33.9	3.01	YES
		6	38	75.3±24.8	-1.50	NO
		7	75	1.0±2.0	36.90	YES
		8	372	77.4±36.7	8.02	YES
		9	172	1.3±2.5	67.10	YES
		10	14	0.0±0.0	INF.	YES
		11	19	0.2±0.6	33.30	YES
		12	13	1.2±1.3	9.38	YES
		13	50	0.0±0.1	501.92	YES
		14	24	0.0±0.0	INF.	YES
		15	6	0.0±0.0	INF.	YES

Table 4.3. Motifs Identification Process; *Final Overall Results*.

Sub-Graph Size	Expertise-level	Motif ID	Significance at the Corresponding Game			
			25%	50%	75%	100%
3	Casual	1	80	100	100	100
		2	80	100	100	100
	Intermediate	1	90	100	100	100
		2	90	100	100	100
	Advanced	1	96.667	100	100	100
		2	96.667	100	100	100
4	Casual	1	86.667	100	100	100
		2	76.667	100	100	100
		3	66.667	100	100	100
		4	10	3.3333	13.333	23.333
		5	83.333	100	100	100
		6	96.667	93.333	93.333	100
	Intermediate	1	90	100	100	100
		2	90	100	100	100
		3	56.667	100	100	100
		4	3.3333	10	20	13.333
		5	90	100	100	100
		6	90	96.667	100	100
	Advanced	1	100	100	100	100
		2	100	100	100	100
		3	90	100	100	100
		4	6.6667	20	30	10
		5	86.667	100	100	100
		6	86.667	100	96.667	100
5	Casual	1	0	23.333	53.333	86.667
		2	60	96.667	100	100
		3	3.3333	3.3333	3.3333	0
		4	53.333	100	100	100
		5	10	33.333	80	93.333
		6	3.3333	0	0	10
		7	43.333	93.333	100	100
		8	23.333	96.667	100	100
		9	53.333	96.667	100	100
		10	0	10	40	96.667
		11	10	63.333	100	100
		12	10	40	66.667	86.667
		13	23.333	73.333	96.667	100
		14	3.3333	13.333	40	100
		15	10	23.333	46.667	83.333
	Intermediate	1	6.6667	46.667	83.333	96.667
		2	70	100	100	100
		3	3.3333	3.3333	3.3333	0
		4	80	100	100	100
		5	16.667	33.333	86.667	90
		6	0	0	3.3333	16.667
		7	50	96.667	100	100
		8	23.333	96.667	100	100
		9	76.667	100	100	100
		10	13.333	30	63.333	90
		11	20	86.667	96.667	100
		12	13.333	66.667	90	93.333
		13	43.333	90	100	100
		14	16.667	40	73.333	90
		15	23.333	50	83.333	96.667
	Advanced	1	20	46.667	73.333	93.333
		2	90	100	100	100
		3	10	3.3333	3.3333	0
		4	100	100	100	100
		5	16.667	63.333	93.333	100
		6	3.3333	0	0	0
		7	70	100	100	100
		8	33.333	100	100	100
		9	80	100	100	100
		10	0	20	56.667	100
		11	30	86.667	100	100
		12	40	73.333	96.667	100
		13	30	66.667	96.667	100
		14	10	30	66.667	100
		15	20	40	66.667	96.667

After explaining the classes of network motifs and the nature of their occurrences in Go – *which constitute the recurring-patterns assessment component (AC_P)*, Figure 4.3 describes the remaining processes of the proposed computational framework in relation to the basic steps of data-mining. Those remaining processes are detailed in the following Sections 4.3 and 4.4.

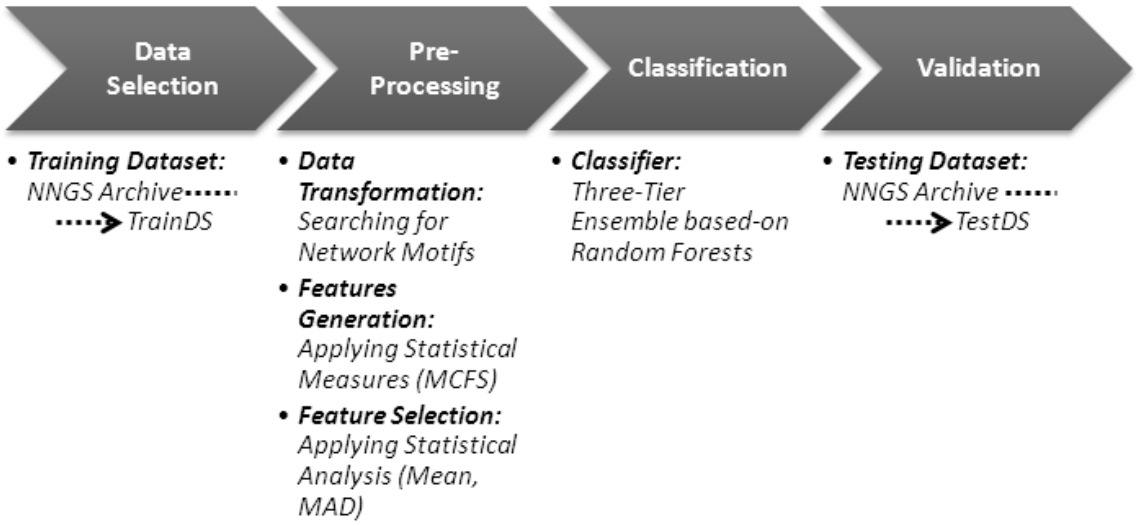


Figure 4.3. An overall illustration of the basic processes for assessing the competency of Go players based on Connectivity-Patterns; detailed in Sections 4.3 and 4.4.

4.3 Estimation of Competency Models

This section presents the first phase of the methodology, in which we will run our system using the *Training Dataset*, which consists of games played by casual, *intermediate*, and *advanced* human opponents. AC_P analysed the games and found the motifs within all of the moves, thus estimating the connectivity-patterns originally employed by its human players. An initial pre-processing step was then carried out by applying a cumulative frequency measurement. This measurement represents the final value that was generated for each game, and thus, the final features' set for the entire games representing a level of expertise. This step was followed by the final statistical analysis in which measures of dispersion and tests of significance were used to examine the implications of the final features sets.

4.3.1 Training Dataset

As previously discussed in Section 2.1, the game of Go traditionally uses the ranking (rating) system of kyu and dan ranks. In this thesis, players with ranks ranging from *30* to *20* kyu are collectively referred to as *Beginners*, ranks from *19* to *10* kyu are *Casual* players, *9* to *1* kyu are *Intermediate* players, and finally from *1* to *7* dan are *Advanced* players. Due to some ambiguities in defining the *Professional* dan ranks in the game records, we decided not to include those ranks in the analysis. This collective reference to the individual kyu-dan ranks assists in minimizing the overlapping between the investigated categories, and in utilizing the very small number of game records remaining after applying our strict conditions for selecting the training dataset.

For the experiments, game records in the text only *smart game format (SGF)* (Hollosi, 2009) were selected from an online NNGS Go Server game archive (Adam, 2009). NNGS *"No Name Go Server"* was a real-time Go server where thousands of online games were played and eventually archived. The server was operating until mid-2005. The cases 'game records' are selected from the years spanning from *1995* to *2005*.

The *training dataset (trainDS)* is one of two datasets that were selected separately, the other being the *testing dataset (testDS)*; from which we will select a set of Go players to observe and monitor their competency. While selecting *trainDS*, the following conditions were taken into consideration:

1. Only standard game records were selected; records representing complete games *(i.e., not stored problems)* played using the standard *19×19* board *(i.e., other board sizes were ignored).*

2. In order to simplify the analysis, only games where both players were from the same ranking range *(e.g., Beginner, Casual … etc.)* were selected.

3. Only games where both players have an established rank are selected; if one or both players were unranked or had an unconfirmed rank, the game was ignored.

4. Only games without handicap were selected, so as not to affect—as much as possible—the application of the typical opening-patterns that are played at the beginning of the game *(fuseki patterns)*.

5. Since games with handicap were not selected, we only looked at relatively equitable games where the difference between the two players was less than 4 stones.

6. To assure—as much as possible—that games were played to the end, only games won by moku *"i.e., points of territory"* are selected; games won by resign, time, or forfeit are ignored. In addition, only games with results were selected; cases for a jigo *"draw"*, no result, suspended play, or unknown result were all ignored.

7. A maximum of five games per registered-name was imposed to limit any personal influences. This rule was made under the assumption that a single player is not likely to be listed using more than a single registered-name.

8. Games with a player's registered-name implying a non-human player *(i.e., an engine)* were discarded; for instance, names such as *robot*, *engine*, *Cshell*, *Gnu*, *manyfaces*, etc.

9. In order to minimize the 'strategical' overlapping between the different rank-categories, only games with both players from the mid-range of a rank category were accepted. The mid-range intervals for the available categories are: *Beginners* [27, 23] kyu, *Casual players* [16, 13] kyu, *Intermediate players* [6, 4] kyu, and *Advanced players* [3, 5] dan.

10. An equal number of games were selected for each rank-category.

Due to these strict conditions, from more than *435,000* games, the final *trainDS* includes *381* games, with *127* games for each category *(Casual, Intermediate, and Advanced players)*. The reason for why no *'Beginner'* cases matched our criteria, is most likely that it usually takes time – *and possibly a certain number of played games* – for a player to have an established rank on a server, thus by that time, nearly every beginner would have already advanced to the casual players range.

4.3.2 Data Pre-processing

AC_P lists the counts of the motifs for each move per player per a game of Go. The main feature then calculated from this counts-list was the following measurement:

- Motifs Cumulative Frequencies per Step *(MCFS)*; for each motif, per player per game, the summation of its corresponding frequencies per step, for all the steps (i.e., moves) of that game. Given the set of all motifs *M*, *MCFS* is defined as:

$$MCFS_{ijk} = \sum_{s=1}^{N_{steps}(i)} C(s,k,M_j)/s \qquad \text{(Eq. 4.2)}$$
$$i = 1 \ldots N_{games},\ j = 1 \ldots N_{motifs},\ k = \{black, white\}$$

where the function *C* returns the – *non-negative integer* – count of the motif M_j in the network generated at the step/move *s* by connecting the stones of the player *k*. N_{games} and N_{motifs} are respectively the total number of games and motifs available, while N_{steps} is the number of moves per game.

4.3.3 Statistical Analysis

After defining the measure to be calculated, the final step is to analyse the grouped measurements – *for the set of games representing a level of performance* – using simple statistical measures *(the Median and the Median Absolute Deviation 'MAD')*, followed by statistical hypothesis testing. Given the set of experiences $E = \{e_1, e_2, \ldots e_n\}$, let D_e denote a subset of the dataset of all games *D*, where the experience of both opponents is *e*. The median can be estimated as:

$$Median_{e,s,\varphi} = \left(\varphi_{\lceil (|D_e|+1)/2 \rceil : |D_e|, M_s} + \varphi_{\lfloor |D_e|/2 \rfloor + 1 : |D_e|, M_s}\right)/2 \qquad \text{(Eq. 4.3)}$$

where φ is the measurement function *(i.e. denoting MCFS)*, M_s is the s*th* motif, $|D_e|$ is the number of games in D_e, and $\varphi_{1:|D_e|} \ldots \varphi_{|D_e|:|D_e|}$ are the order statistics of $\varphi_1 \ldots \varphi_{|D_e|}$. Accordingly, MAD can be estimated as:

$$MAD_{e,s,\varphi} = Median\left(\begin{array}{l}\left|\varphi_{1:|D_e|,M_s} - Median_{e,s,\varphi}\right|,\ldots \\ \ldots,\left|\varphi_{|D_e|:|D_e|,M_s} - Median_{e,s,\varphi}\right|\end{array}\right) \qquad \text{(Eq. 4.4)}$$

Our main argument is that the medians of the different motifs-counts can model how the general strategy is decomposed into characterizing connectivity-patterns, in addition to demonstrating the variations in those patterns employed by the human Go players of different experiences. To confirm the potential hypotheses suggested by the data, both a two-sample T-test and a two-sided Wilcoxon rank sum test are used.

The two-sample T-test tests a null hypothesis H_0 that the two independent samples come from normal distributions with unknown variances and the same mean, against the alternative that the means are unequal. The test is two-tailed, and performed at a significance level $\alpha = 0.05$, i.e. the probability of mistakenly rejecting H_0 *(Type I error)* is no more than *5%*. Alternatively, the Wilcoxon-test tests a null hypothesis H_0 that the two independent samples come from identical continuous distributions with the same median, against the alternative that the medians are unequal. The Wilcoxon-test is also performed at a significance level $\alpha = 0.05$. By permuting the types of Motifs, the calculated measurement, and pairs of different experiences, the T-test and Wilcoxon-test will examine the null hypothesis that the data *(i.e. the MCFS per game)* are with equal means/medians 'respectively' against the alternative that the means/medians are not equal.

4.3.4 Discussion

In the first phase of the experiments, the system was run using the selected case-base of games that were played by *casual*, *intermediate*, and *advanced* human players. The motifs were then counted for all of the moves, thus assessing the connectivity-

patterns originally employed by the human players. The Motifs Cumulative Frequencies per Step (Eq. 4.2) were applied as measurements for grouping the counted motifs per game. Subsequently, and between each distinct pair of experiences, the Wilcoxon-test and a two-sample T-test were applied to statistically signify the ability of the calculated medians/means to differentiate between the corresponding distinct pair of experiences. Table 4.4 shows the medians (Eq. 4.3) and median absolute deviations (Eq. 4.4) among the *127* games per experience level.

Almost half of the motifs failed in significantly differentiating between the expertise-levels; either by completely failing to differentiate between any pair of the expertise-levels, or differentiating between only one pair. Table 4.4 shows that, *11* motifs – *out of the 20 investigated* – showed a statistical difference between at least *2* pairs of expertise-levels. Mostly, the capability was to discriminate between casual/intermediate and casual/advanced human Go players, while failing to discriminate statistically between intermediate/advanced players. However, M4-2 was found capable of discriminating between all three pairs of expertise-levels, and M5-15 was capable of discriminating between Intermediate players on one side, and both Casual and Advanced players on the other. The two latter observations had not been paralleled by any of the reasoning sub-sets investigated in Chapter 5.

In general, the medians reported for the Intermediate players tend to be the highest, followed by the Advanced players, and finally the Casual players. On the other hand, the Causal players reported the highest Median Deviations, followed by the Intermediate players, while the Advanced players reported the lowest deviations. This latter observation shows that the Advanced-players' playing capabilities are more conformed when compared to other performance levels, a conclusion that had also been supported by the future findings in Section 5.3.4.

Finally, from the analysis of the reported results, the detected connectivity-patterns within the proposed methodology were adequately capable of characterizing particular differences in the expertise levels – of the human Go players – in an

automated and passive approach, the description of these differences was quantitatively-represented and statistically-significant. However, only a subset of the investigated motifs was able to provide a significant discrimination. For an additional demonstration of those Motifs, Figure 4.4 will illustrate the *11* finally-selected Motifs through a representative Go game.

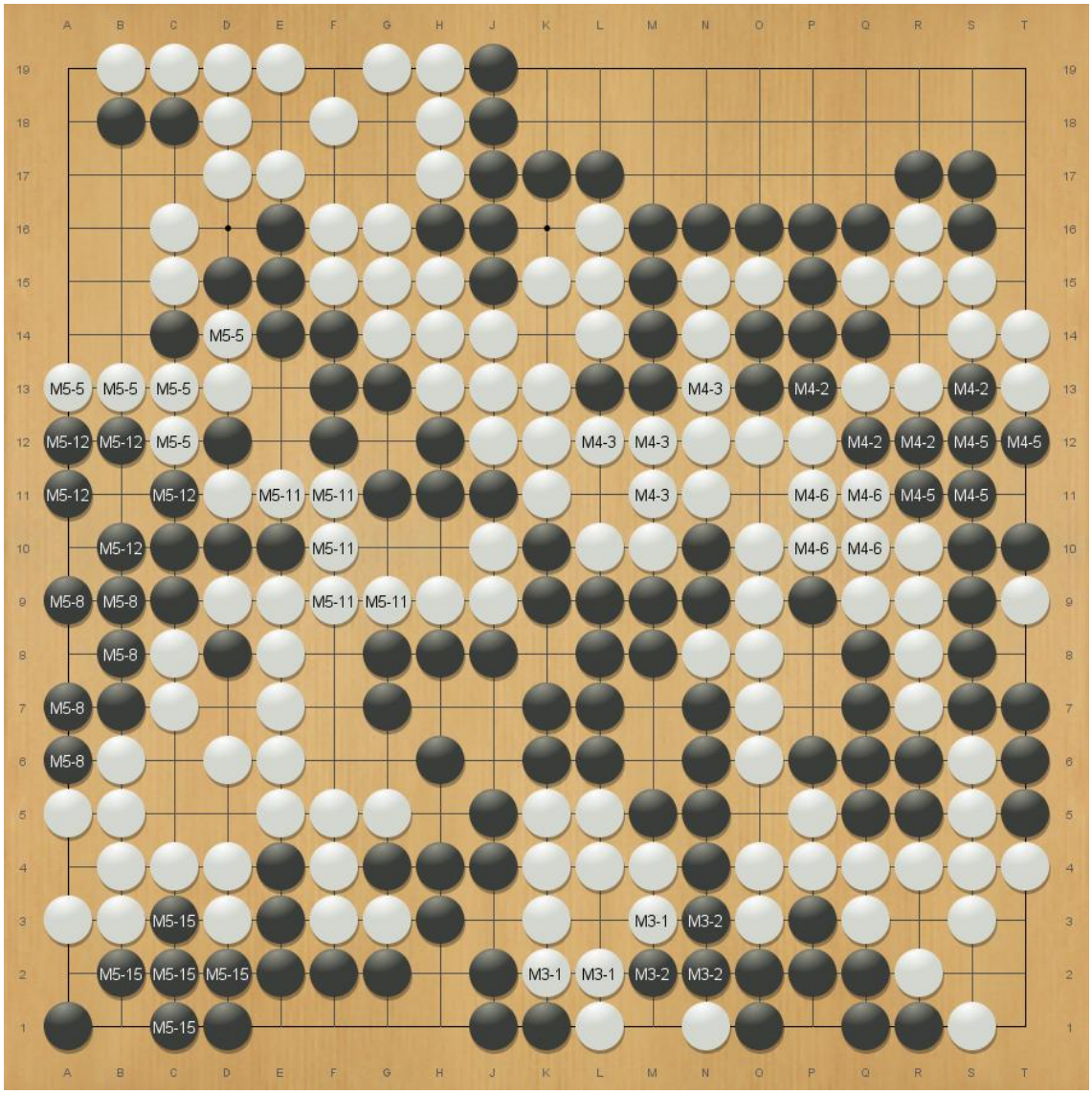


Figure 4.4. The *11* Finally selected motifs illustrated through a representative Go game. The *2* motifs of size *3* are shown at the bottom of the figure. The *4* motifs of size *4* are shown at the right-hand side of the figure, while the *5* motifs of size *5* are shown at the left-hand side of the figure.

Table 4.4. The Medians and Median Absolute Deviations (MAD) of the Different Motifs among Diverse Experience Levels.

		Level of Expertise					
		Casual		**Intermediate**		**Advanced**	
Sub-Graph Size	**Motif ID**	**Median**	***MAD***	**Median**	***MAD***	**Median**	***MAD***
3	**1** $^{\alpha}$	131.62	*33.6*	157.7	*32.884*	144.77	*31.418*
	2 $^{\alpha}$	27.325	*8.7859*	32.486	*8.3568*	31.173	*7.8946*
4	**1**	15.395	*7.5363*	17.585	*7.3493*	17.275	*6.5378*
	2 $^{\beta}$	151.96	*49.894*	183.08	*49.093*	164.94	*46.069*
	3 $^{\alpha}$	54.365	*22.122*	69.807	*21.837*	64.62	*20.362*
	5 $^{\alpha}$	13.845	*6.5983*	16.456	*6.2237*	15.604	*6.025*
	6 $^{\alpha}$	0	*0.23486*	0.028124	*0.30052*	0.027031	*0.28119*
5	**1**	0.42811	*0.7384*	0.56672	*0.69504*	0.63813	*0.68844*
	2	59.183	*32.014*	72.819	*28.422*	66.94	*26.302*
	4	193.39	*77.875*	230.86	*72.563*	209.49	*67.04*
	5 $^{\alpha}$	34.535	*17.131*	43.245	*17.028*	40.243	*15.542*
	7	12.105	*6.8969*	13.956	*6.5749*	13.417	*5.9627*
	8 $^{\alpha}$	79.722	*35.032*	93.683	*33.722*	88.6	*32.045*
	9	30.997	*16.099*	36.099	*14.379*	34.375	*14.231*
	10	0.69341	*0.96935*	0.82733	*1.094*	0.70041	*0.98051*
	11 $^{\alpha}$	4.8236	*2.7678*	5.8305	*2.9316*	5.7142	*2.5667*
	12 $^{\alpha}$	2.3781	*2.1515*	2.8842	*2.1432*	2.9095	*2.0222*
	13	6.6227	*4.3188*	8.0854	*3.9196*	7.5387	*3.8733*
	14	1.5888	*2.3297*	1.8693	*2.1369*	1.7353	*2.2676*
	15 $^{\gamma}$	0.68753	*0.79779*	0.45382	*0.60478*	0.71075	*0.67814*

All the values reported in this table are calculated using *127* games per each level of experience.

Light grey rows represent motifs that fail to differentiate between at least 2 pairs of expertise-levels.

$^{\alpha}$ The rows where the motif can, in a statistically significant manner, differentiate between Casual/Intermediate and Casual/Advanced.

$^{\beta}$ The rows where the motif can, in a statistically significant manner, differentiate between all three pairs of expertise-levels.

$^{\gamma}$ The rows where the motif can, in a statistically significant manner, differentiate between Casual/Intermediate and Intermediate/Advanced.

4.4 A Case Study: Tracking the Motifs-Based Competency

In the second and final phase of our experiments, we created our proposed classifier. Then, using *testDS*, the games of each player were temporally ordered and classified. The resulting classification probabilities have been cumulatively averaged to generate *Monitoring-Curves* to observe the learning activity per player. This leads to the final stage, in which this learning activity was diagnosed by temporally observing each of the motifs' characteristics.

4.4.1 Classifier Architecture

An idea proposed in the literature since the late sixties, and matured in the early nineties, is of whether by combining a set of varying Machine Learning classifiers *(classifier ensembles)*, can the overall performance be improved in comparison to a single classifier. Thus instead of relying on single classifiers to solve a specific domain-related task, Ensemble Learning (EL) aims to aggregate a set of classifiers that vary in their nature, structure, or even their operational parameters, while exploiting the hope that *"the sets of patterns misclassified by the different classifiers would not necessarily overlap"* (Kittler et al., 1998). Since then, classifier ensembles have been applied to a wide range of domains, and various methods for combining classifiers have been investigated and evaluated in the literature.

The accuracy of Random Forests, just as any other ensemble model, depends mainly on the individual-strengths and diversity of the base models. Therefore, ensembles are effective when applied to unstable classifiers *(e.g., classification trees)*; in the case of which, any minor change in the operational parameters or the training set ends up with a noticeable change in the classifier's prediction. Besides bagging *"**B**ootstrap **Agg**regat**ing**"* in which new training sets *'bootstrap replicas'* are drawn with replacement from the original set, randomness is also introduced to the forests by random feature selection, whereby a random subset of features *'input variables'* is used at each decision *'node'* split.

Since bagging omits on average one third of the observations in each replica, those omitted *(so-called out-of-bag)* observations can be used to estimate the error in the bagged classifiers. This is done for each observation by averaging the predictions of all the ensemble's trees in which this observation is out-of-bag. The out-of-bag estimate is unbiased, and is practically as accurate as using a testing set of a size similar to the training set (Breiman, 2001). Using random features makes the forests relatively more robust to noise and faster than bagging alone, characteristics that we may need when classifying features of which some are potentially irrelevant. For a detailed description of random forests, the reader may refer to (Breiman, 2001).

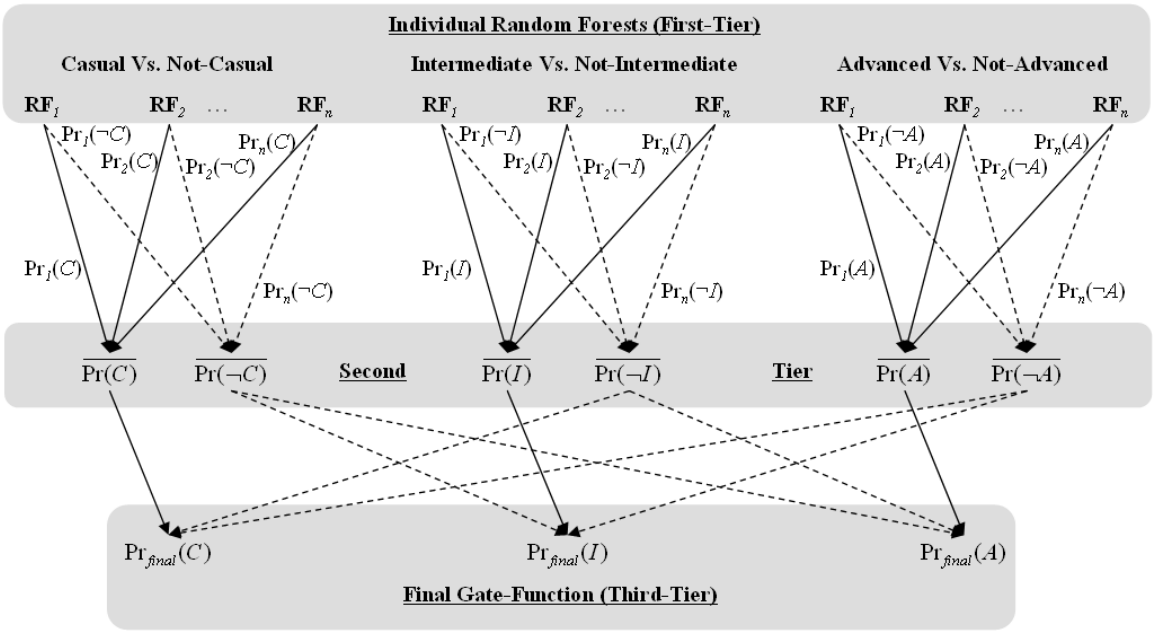


Figure 4.5. The three-tier ensemble used to predict the class label of a game of Go.

In this study, a three-tier ensemble (Figure 4.5) is used to predict the class label of a game of Go as *Casual*, *Intermediate*, or *Advanced*. The first-tier is based on Random Forests (RFs), which are ensembles of Classification Decision Trees (CDTs). In order to analyse the motifs counts, we are looking for a robust white-box model, which can handle data without requiring a lot of data preparation. These requirements suggest the use of CDTs. For this, each individual classifier (i.e., RF) is trained to classify a class and its complement; for example, a RF is trained to classify *Casual*

games versus *Not-Casual* (i.e., *Intermediate* and *Advanced*) games, and so on. Thus, each RF outputs two probabilities 'Pr'; namely for the previous example, a probability that a given game is originating from the *Casual* class: Pr(*C*), and a probability that the same given game is originating from the *Not-Casual* class*:* Pr(¬*C*). The RF is created using a maximum number of decision trees (set to *1000* throughout this study), yet only the number of trees corresponding to the minimum cumulative misclassification probability—for out-of-bag observations in the training data—is used while testing (Figure 4.6).

The second-tier creates an ensemble of RFs (i.e., a Forest of Random Forests) for each class which then averages their predicted probabilities over all RFs in the ensemble. For instance, the *Casual Forest* is an ensemble of *n* RFs which are each trained to classify *Casual* games versus *Not-Casual*. The resulting averaged probabilities are given as follows:

$$\overline{\Pr(Class_j)} = \sum_{i=1}^{n} \Pr_i(Class_j) \Big/ n \qquad \text{(Eq. 4.5)}$$

and

$$\overline{\Pr(\neg Class_j)} = \sum_{i=1}^{n} \Pr_i(\neg Class_j) \Big/ n \qquad \text{(Eq. 4.6)}$$

$$with\ \ j = 1 \ldots N_{classes}$$

where $N_{classes}$ is the number of available classes (i.e., experience-levels). Throughout this study, *n* is set to *30*. The third and final tier combines the results from the second-tier forests using a final gate-function to create—for each observation (i.e., game)—a single probability 'Pr_{final}' per class. The final gate-function is given as follows:

$$\Pr_{final}(Class_j) = \left[\overline{\Pr(Class_j)} + \sum_{i=1,\, i \neq j}^{N_{classes}} \overline{\Pr(\neg Class_i)} \right] \Big/ N_{classes} \qquad \text{(Eq. 4.7)}$$

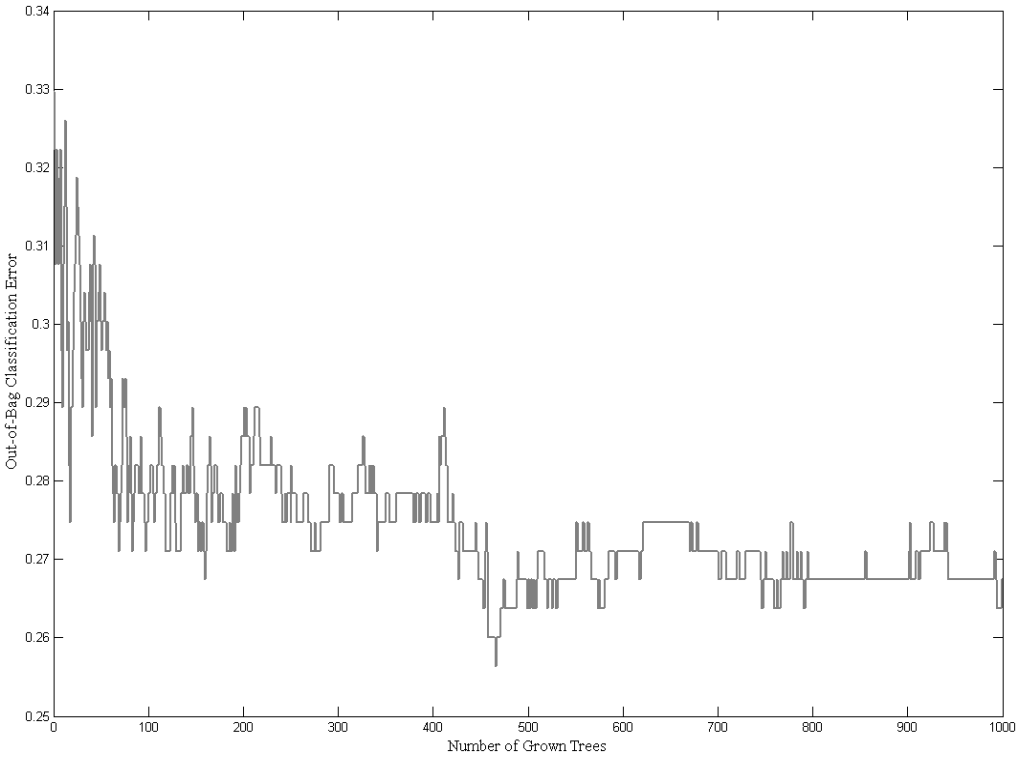


Figure 4.6. The cumulative out-of-bag classification error for a *1000*-tree RF trained to classify *Casual* versus *Not-Casual* games using the Reasons Frequencies per Step *(FS)*. The minimum out-of-bag classification error is *0.2564*, and the corresponding number of trees is *466*.

4.4.2 Testing Dataset: The Selection of Cases

The *testDS* is used to select the cases to be monitored, thus the strict conditions used for selecting the *trainDS* should be relaxed in order to the allow for a bigger number of games having common players; in other words, allowing for players with sufficient history. So, the *trainDS* selecting-conditions numbered *2*, *4*, *7*, *9*, and *10* were discarded.

Four hundred games were then selected for the *testDS*—to be comparable in size with the *trainDS*—with only *16* games found to be common between the two datasets. The *400* games were played by *246* distinct registered-names, thus—to select the final

cases—a threshold of *10* games at least was imposed, and this yielded a final set of *15* players (Table 4.5).

Table 4.5. The final *testDS* representing the *15* players to be observed.

	Registered Names	**Number of Games Played**	**Number of Games that are Common between Both Datasets**	**The Averaged[1] Experience Range Covered by the Corresponding Player's Games**
1	Ccwills	*74*	*5*	*Upper-Beginner to Lower-Intermediate*
2	Giv	*38*	-	*Lower-Intermediate to Mid-Intermediate*
3	Tarok	*32*	*1*	*Mid-Intermediate*
4	Teto	*46*	*5*	*Mid-Beginner to Mid-Casual*
5	Ggl	*20*	-	*Lower-Intermediate*
6	Bommeltk	*16*	*2*	*Mid-Casual to Lower-Intermediate*
7	LordOfPi	*35*	*5*	*Mid-Casual to Upper-Casual*
8	Gerula	*13*	-	*Mid-Casual to Upper-Casual*
9	YHH	*36*	-	*Upper-Intermediate*
10	Mattias	*26*	-	*Lower-Intermediate to Mid-Intermediate*
11	Koisan	*50*	-	*Lower-Advanced*
12	Kouichi	*11*	-	*Lower-Advanced*
13	Shouhei	*57*	-	*Upper-Intermediate to Lower-Advanced*
14	Faramund	*10*	-	*Mid-Intermediate*
15	Uly	*34*	-	*Lower-Intermediate to Mid-Intermediate*
Total[2]		***498***	***18***	

[1] The Experience Range covered is an un-weighted cumulative moving average—of the corresponding player's online rank—with a maximum window size of *50*. See Section 6.3 for more details.

[2] Since there are games where both opponents are from the final selected players, the total values of both columns—*498*, and *18*—are respectively higher than the total number of games in *testD* (*400*) and the total number of games found to be common between both datasets (*16*).

4.4.3 Experimental Setup

In the conducted experiments, RFs were used which had a maximum of *1000* classification decision trees, and a minimum number of observations per tree leaf of *1*, A forest's attributes were determined according to the *Error* and *Size* measures. These are respectively, the minimum error *(i.e., misclassification probability for the out-of-bag observations)* recorded during the process of adding up trees while creating the forest and the ensemble size *(i.e., number of trees)* corresponding to that minimum error value. Using *trainDS*, *30* random forests were trained to differentiate between each experience level and its complement. Those *30* random forests that were trained – per experience, and using *MCFS* as the measurement – have been combined to form the second-tier ensemble.

4.4.4 Discussion

The RFs training presented in the previous section have been employed to create the final proposed classifier (Figure 4.5). The *30* random forests trained – per experience, and using *MCFS* as the measurement – have been combined to form the second-tier ensemble. Using the *testDS*, the games for each player – among the *15* finally selected – are temporally ordered and then analysed to count the motifs within. *MCFS* is then applied thus creating the final feature set for each game. The proposed classifier generates three final probabilities for each game, namely $\Pr_{final}(C)$, $\Pr_{final}(I)$, and $\Pr_{final}(A)$. Per player, three *Monitoring-Curves* have been plotted; each representing the 'un-weighted' Cumulative Moving Average (CMA) for an experience-level, with a maximum *window size* of *50* games. Given the number of available experience-levels as $N_{classes}$, and the total number of games per single player as N_{games}, the CMA – for the corresponding player – is computed as follows:

$$CMA(Class_j)_i = \frac{\Pr_{final}(Class_j)_k + \ldots + \Pr_{final}(Class_j)_i}{(i-k)+1} \qquad \text{(Eq. 4.8)}$$

$$with\ \ j = 1 \ldots N_{classes},\ i = 1 \ldots N_{games},\ and\ k = \begin{cases} 1 & i \le 50 \\ (i-50)+1 & i > 50 \end{cases}$$

For the several forthcoming pages, a 2-D line graph with two *y*-axes – on both the left and right sides – is presented for each player. The *y*-axis on the left is labelled *'Player's Experience Curves'* and displays the value of the three *Monitoring-Curves*, while the *y*-axis on the right is labelled *'Ranks' Categories'* and displays the *Player's Rank* according to the online *No Name Go Server* archives. The *Player's Rank* curve is also a CMA of the actual rank-values, and uses the same maximum *window-size* of *50*. A label on the right *y*-axis represents the centre of the respective rank category. The *x*-axis displays the game number, with imposed temporal frames for the corresponding dates (months/years).

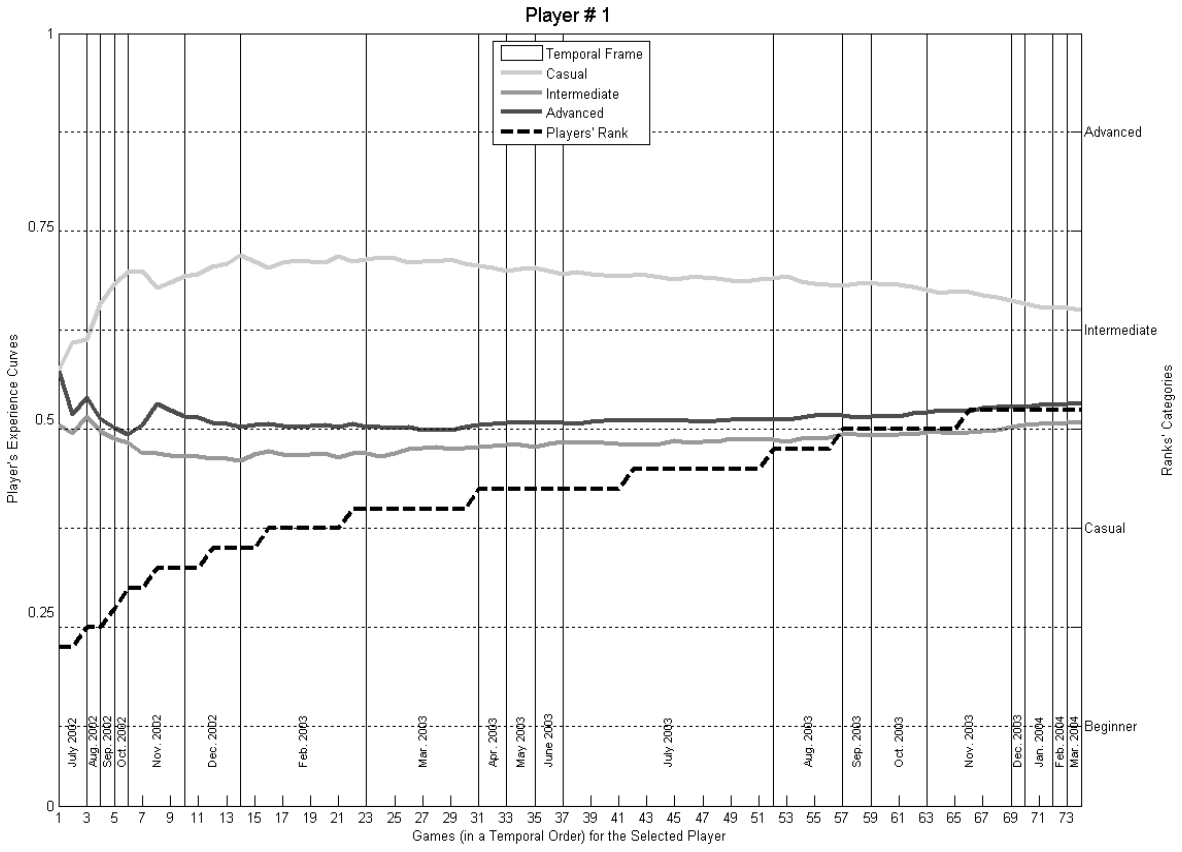


Figure 4.7 (a). The Competency-Level Monitoring Curve – Player #1; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

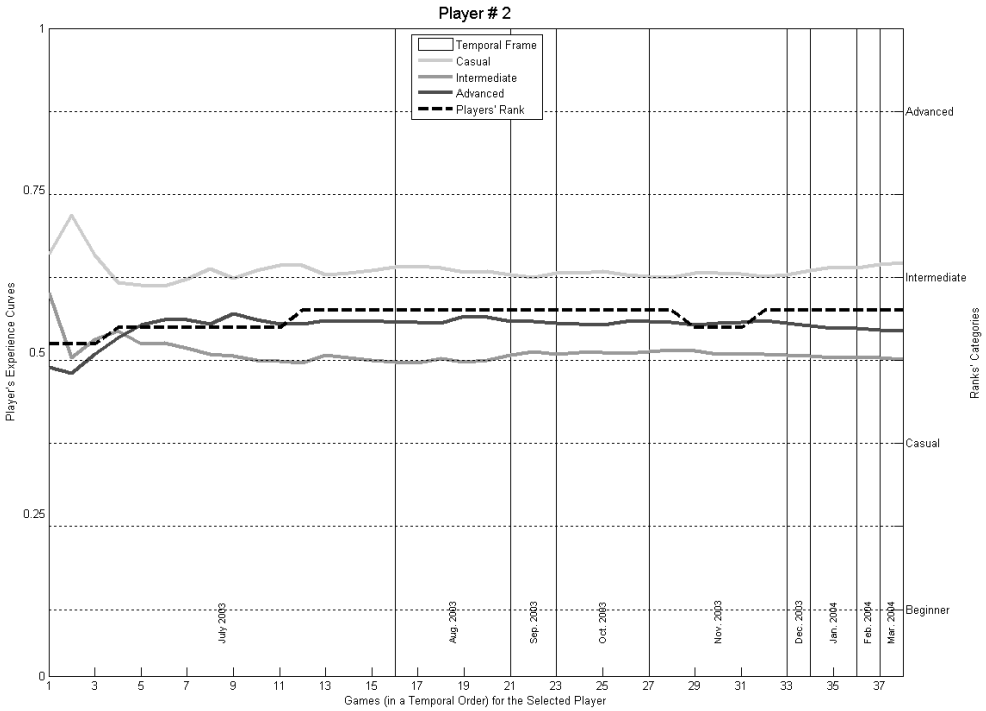


Figure 4.7 (b). The Competency-Level Monitoring Curve – Player #2; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

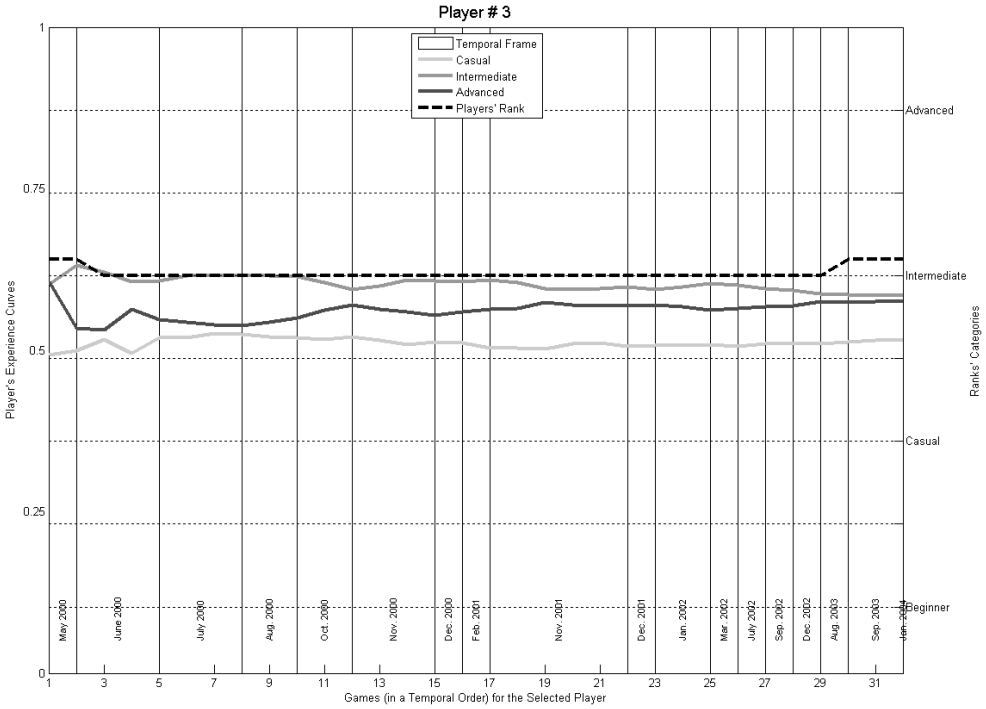


Figure 4.7 (c). The Competency-Level Monitoring Curve – Player #3; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

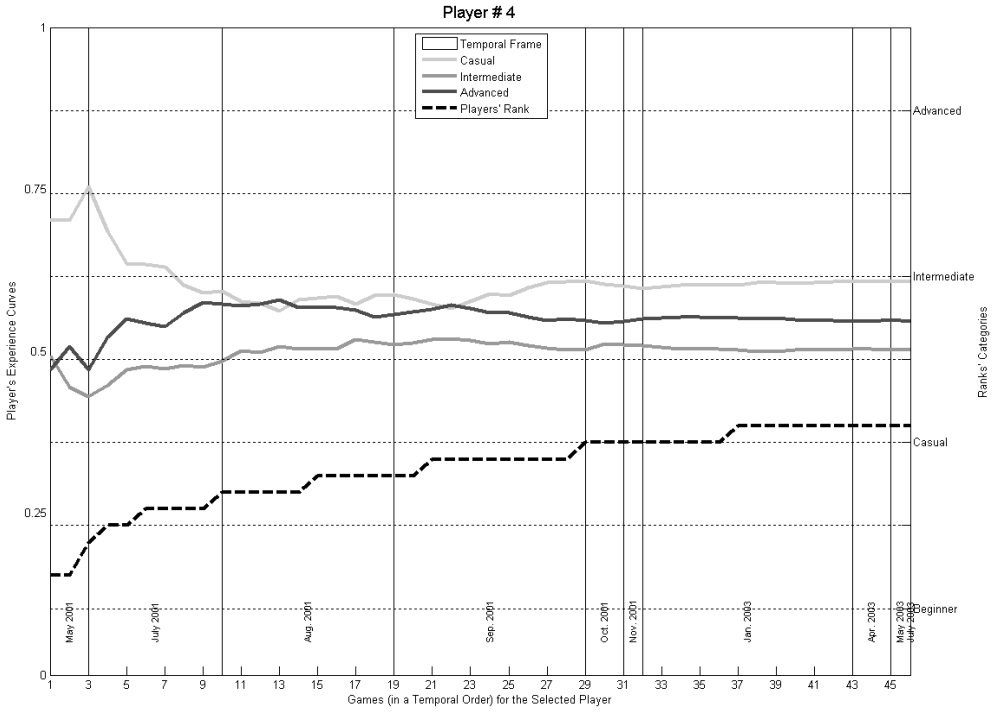


Figure 4.7 (d). The Competency-Level Monitoring Curve – Player #4; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

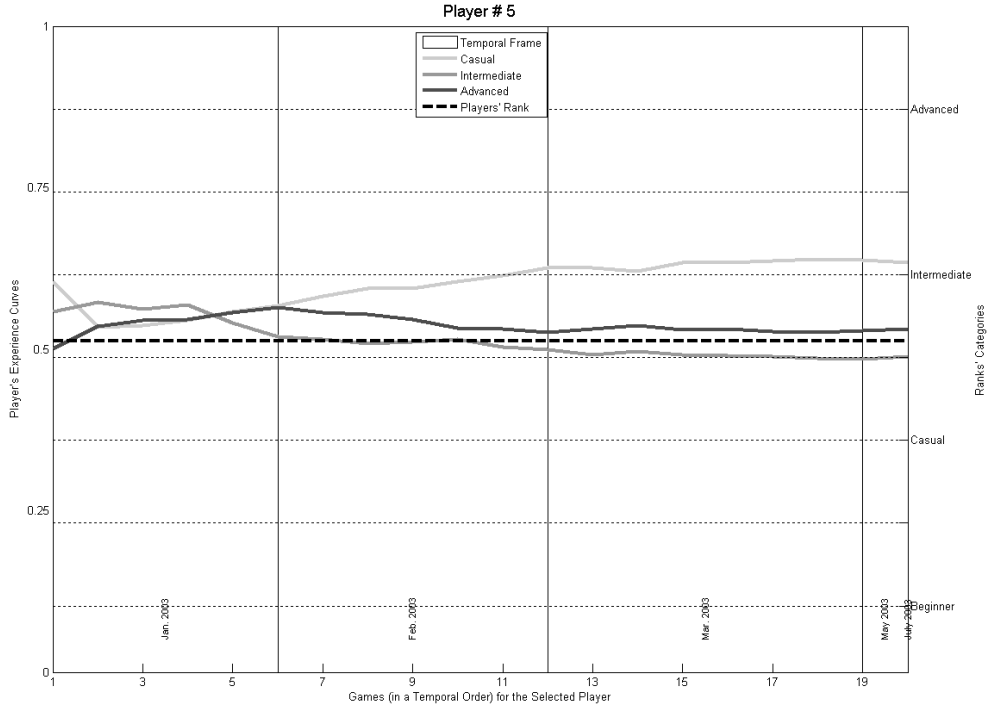


Figure 4.7 (e). The Competency-Level Monitoring Curve – Player #5; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

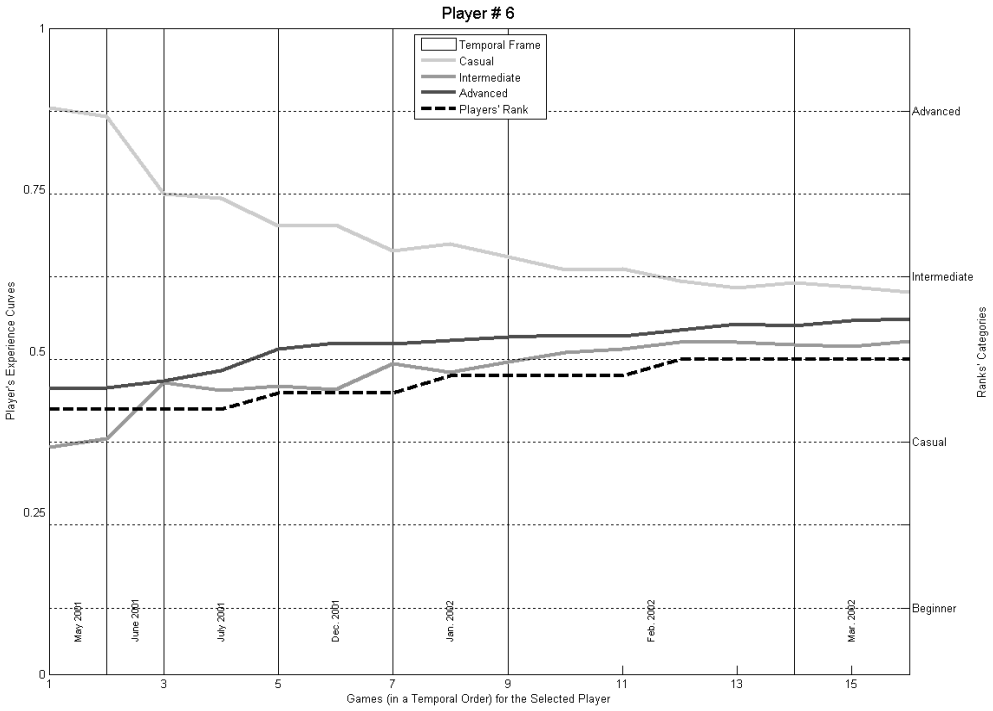


Figure 4.7 (f). The Competency-Level Monitoring Curve – Player #6; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

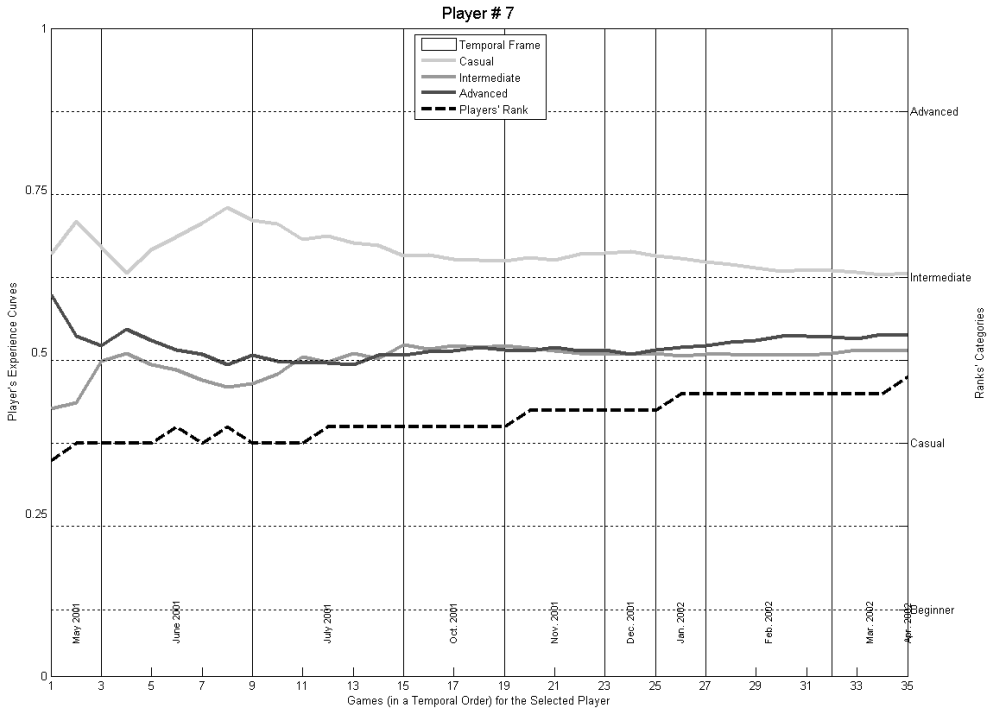


Figure 4.7 (g). The Competency-Level Monitoring Curve – Player #7; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

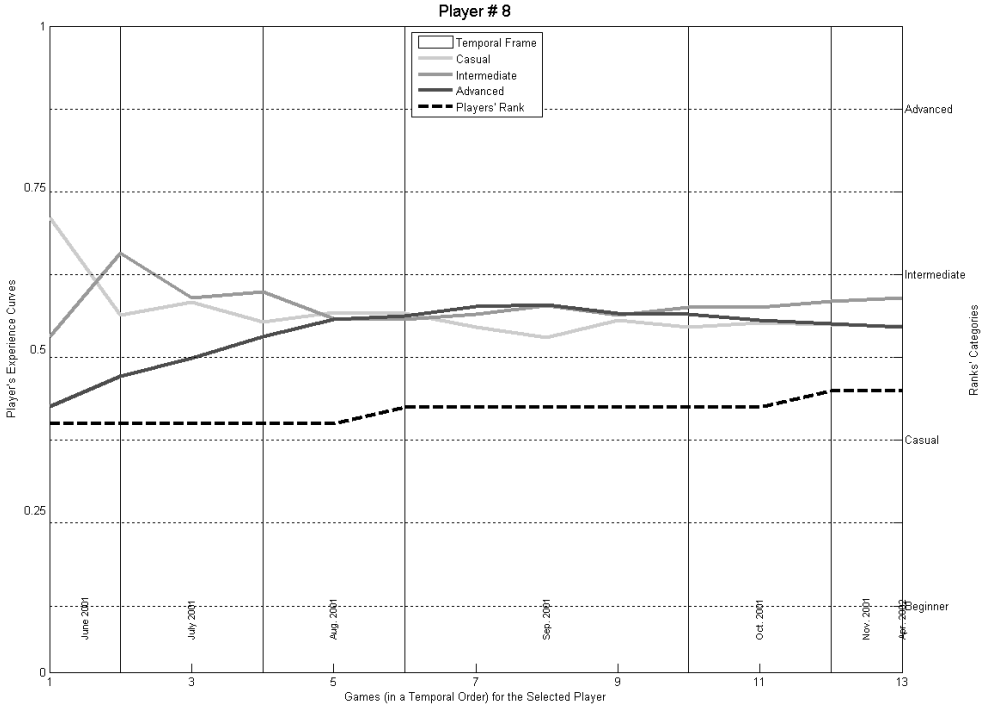


Figure 4.7 (h). The Competency-Level Monitoring Curve – Player #8; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

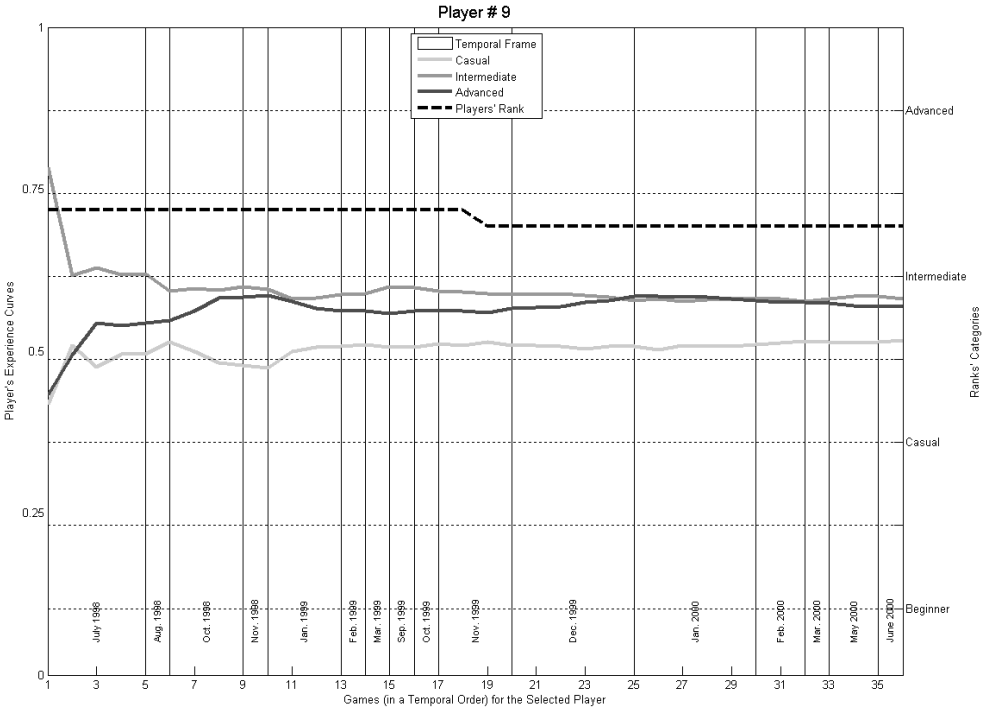


Figure 4.7 (i). The Competency-Level Monitoring Curve – Player #9; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

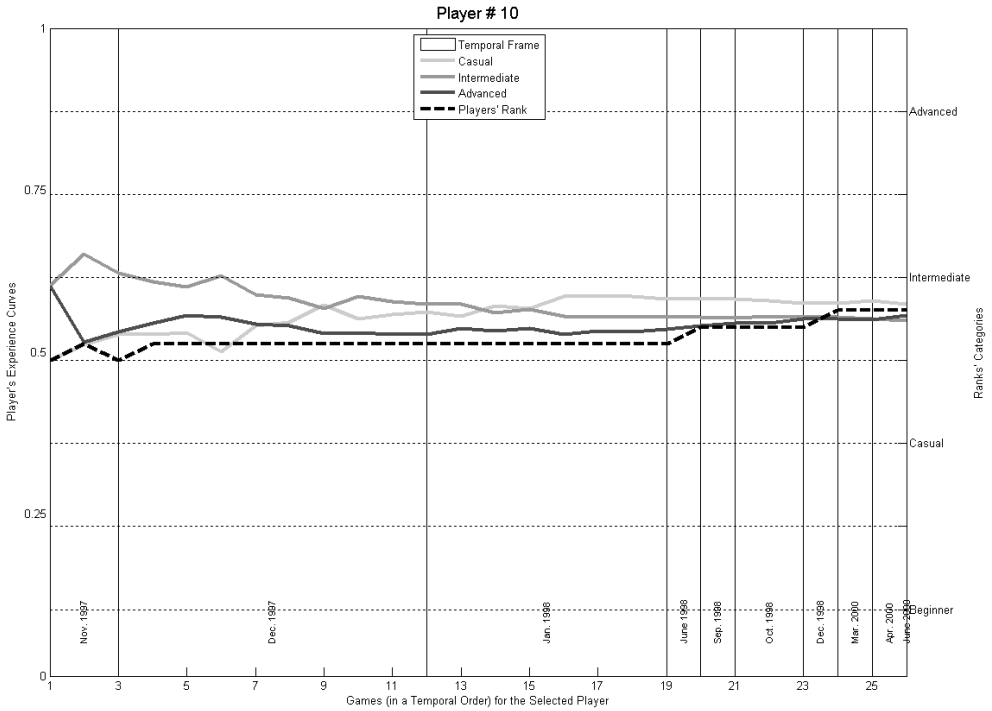


Figure 4.7 (j). The Competency-Level Monitoring Curve – Player #10; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

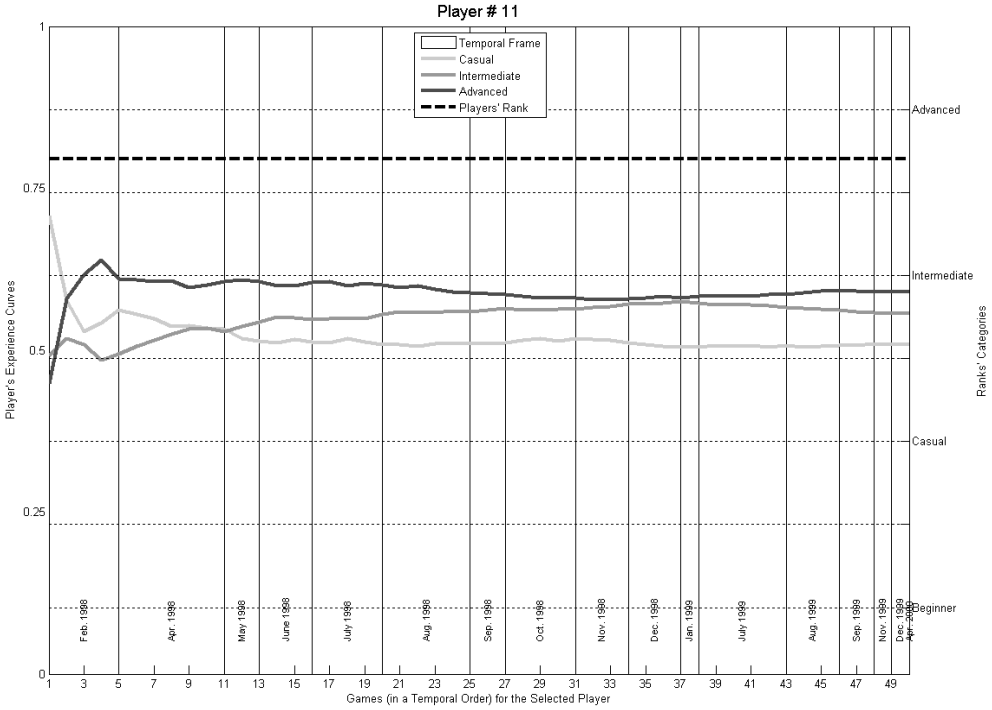


Figure 4.7 (k). The Competency-Level Monitoring Curve – Player #11; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

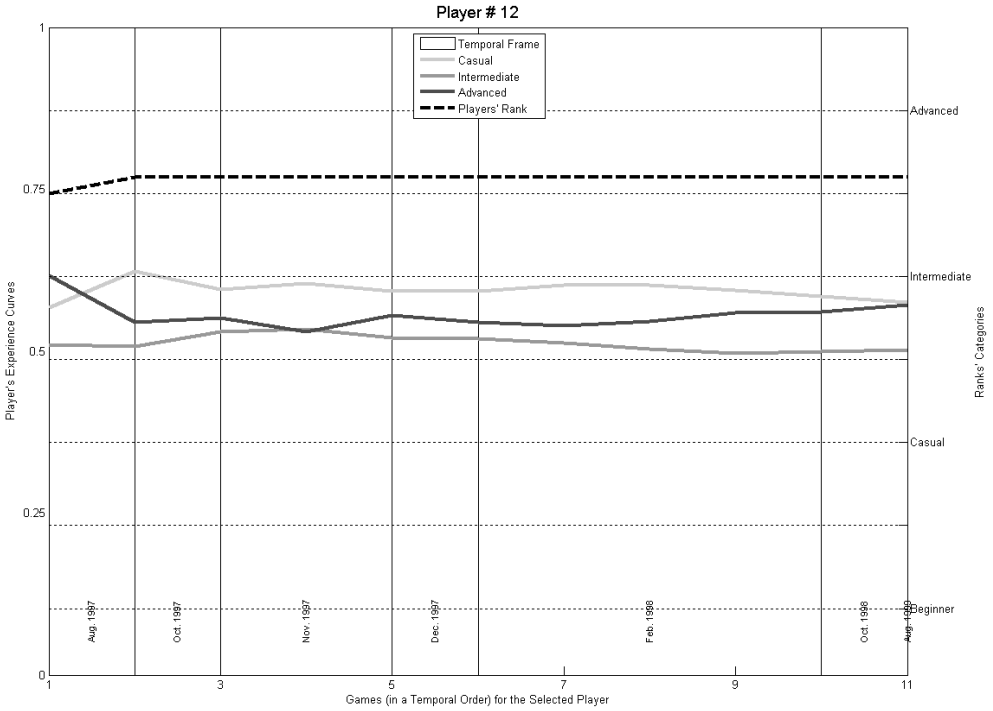


Figure 4.7 (l). The Competency-Level Monitoring Curve – Player #12; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

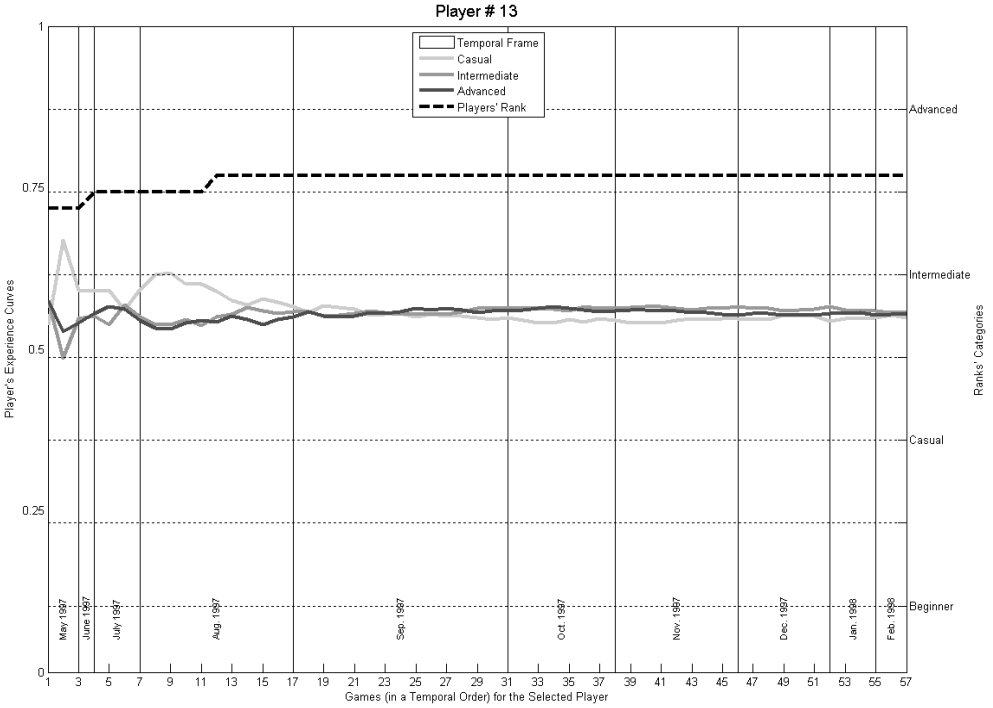


Figure 4.7 (m). The Competency-Level Monitoring Curve – Player #13; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

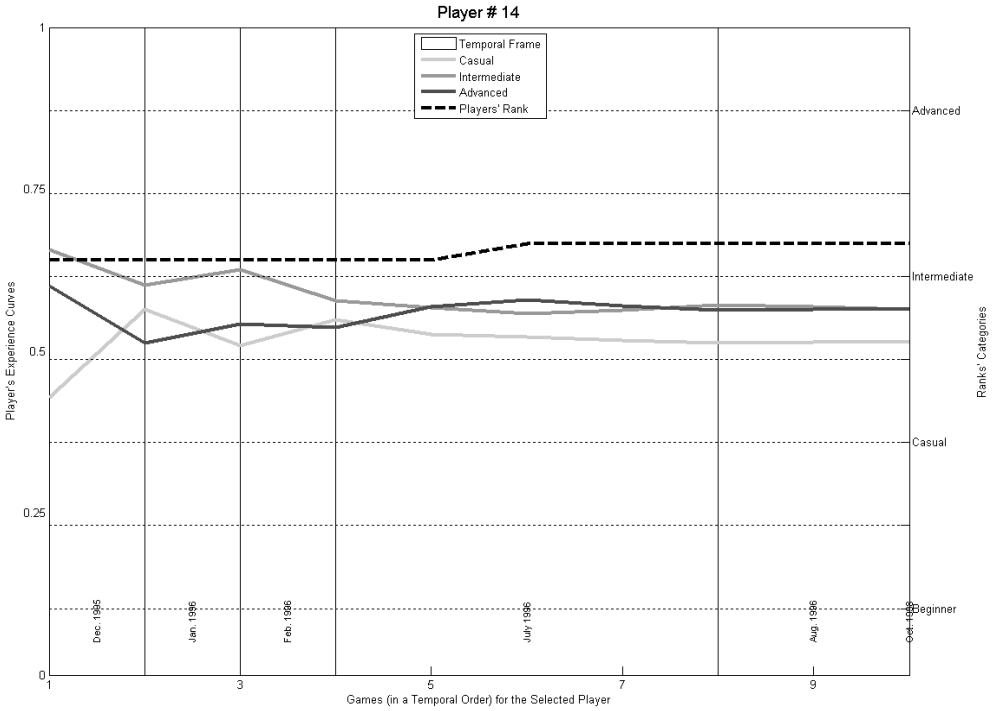


Figure 4.7 (n). The Competency-Level Monitoring Curve – Player #14; *Based on the motifs identification of the individual player and the proposed three-tier classifier.*

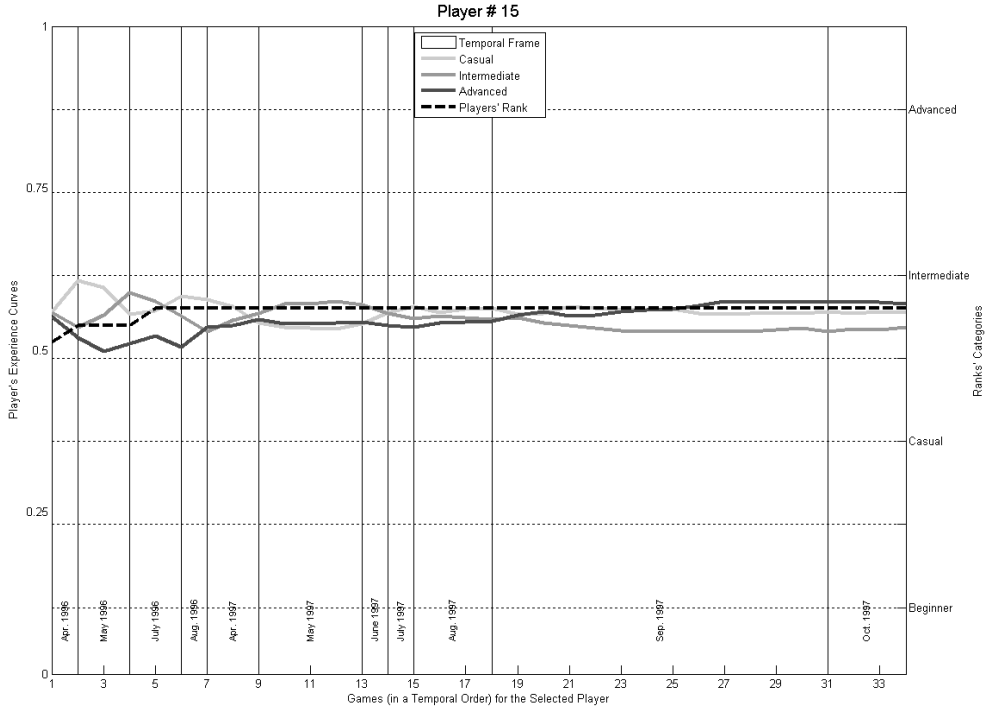


Figure 4.7 (o). The Competency-Level Monitoring Curve – Player #15; *Based on the motifs identification of the individual player and the proposed three-tier classifier*.

In the majority of the figures – *10* out of the *15* players analysed – the resulting figures show an adequate consistency between the experience level of a player and their probabilities' curves. For instance, "Player #1" advances from an Upper-Beginner to a Lower-Intermediate experience over the course of the *74* games selected. Concurrently, this player is classified as a Casual during the whole period. In addition, there is a trend in the curves that assures that – with more games – the player is going to be correctly classified as Intermediate.

Similar to "Player #1", the monitoring curve corresponding to the correct experience level is mostly higher than the other two curves for players *3*, *4*, *6*, *7*, and *11*. However, not all the remaining players are entirely misclassified. For instance, the "Player #9" is one where the *36* games fall in the Upper-Intermediate experience level, and where starting from the 25^{th} game, the Advanced curve is slightly higher than the Intermediate.

Those reported results demonstrate that the "Competency Model" developed using the quantitative measurements representing the motifs counts is capable – *yet less successfully when compared to situational-reasoning in Chapter 5* – of somehow reflecting and monitoring the changes in the performance of a selected individual player. Additionally, as proposed, the monitoring approach is completely automated and passive.

In order to provide a more quantitative interpretation of the competency-curves and their temporal trends with the changing level-of-expertise (i.e., rank), Table 4.6 will show the correlation between each competency-curve and the rank. Correlation is a statistical measure of dependence between two variables; that is, a measure of how strong the relationship between both variables is. Thus, correlation results in a value between *+1* and *-1* inclusive; the former implies a strong positive relationship while the later implies a strong negative relationship. A value of *0* implies the absence of any correlation between both variables.

The correlation is calculated as follows: Given the number of available experience-levels as $N_{classes}$, the total number of games per single player as N_{games}, the 'un-weighted' Cumulative Moving Average (CMA) for monitoring an experience-level j as CMA_j, and the experience-level for a specific player as *Rank*. The correlation coefficient *'r'* for the corresponding player – *per an experience level* – is computed as follows:

$$r_{CMA_j,Rank} = \frac{\mathrm{cov}(CMA_j, Rank)}{\sigma_{CMA_j}\sigma_{Rank}} \qquad \text{with } j = 1 \ldots N_{classes} \qquad \text{(Eq. 4.9)}$$

where *'cov'* is the statistical covariance measure, defined as:

$$\mathrm{cov}(CMA_j, Rank) = \frac{\sum_{i=1}^{N}((CMA_j)_i - \mu_{CMA_j})(Rank_i - \mu_{Rank})}{N} \qquad \text{with } i = 1 \ldots N_{games}, \mu_x = \text{the mean of the values} \qquad \text{(Eq. 4.10)}$$

and where *'σ'* is the standard-deviation, defined as:

$$\sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)^2} \qquad \text{(Eq. 4.11)}$$

Table 4.6 *(Continued in the next page).* The Correlation between the Selected Players' Ranks and the Competency Monitoring Curves *(Based on Connectivity-Patterns).*

Player ID	**Correlation Coefficient: *between the Experience Range and the …***			**The Averaged Experience Range Covered by the Corresponding Player's Games**	**Observations**
	Casual Curve	***Intermediate Curve***	***Advanced Curve***		
1	0.03089	0.43643	0.13694	*Upper-Beginner to Lower-Intermediate*	A moderate correlation with the intermediate curve indicates the acquisition of intermediate skills, besides a slight acquisition of some Advanced skills, and no dependence with the causal curve.
2	-0.40779	-0.56186	0.65999	*Lower-Intermediate to Mid-Intermediate*	A moderate positive correlation with Advanced skills, in addition to moderate negative correlations with both casual and intermediate skills.
3	-0.22466	-0.19633	0.32477	*Mid-Intermediate*	A weak negative correlation with both Casual and Intermediate Curves, and a moderate correlation with the Advanced curve even though the player's rank is not progressing.
4	-0.55909	0.6826	0.43884	*Mid-Beginner to Mid-Casual*	A moderate negative correlation with Casual curve, indicating that the player developing beyond that level. This is supported by positive moderate correlation with Intermediate curve. Yet, a positive moderate correlation with the Advanced curve exists too.
5	No Variation in the Experience-Level, thus no trend is obtainable.			*Lower-Intermediate*	Not Applicable
6	-0.88227	0.85421	0.93664	*Mid-Casual to Lower-Intermediate*	The trend in the Experience Level is clearly shown by strong – respectively negative and positive – correlations with both Causal and Intermediate curves. However, a stronger positive correlation with the Advanced curve exists.

7	-0.64116	0.57293	0.06963	*Mid-Casual to Upper-Casual*	A moderate negative correlation with the Casual curve, accompanied with a moderate positive correlation with the Intermediate curve.
8	-0.4615	-0.10242	0.568	*Mid-Casual to Upper-Casual*	A moderate positive correlation with the Advanced curve that is not mirrored by the experience levels.
9	-0.43919	0.39556	-0.41796	*Upper-Intermediate*	A moderate negative correlation with both Casual and Advanced curves, and a weak positive correlation with the Intermediate curve.
10	0.53952	-0.57836	0.1028	*Lower-Intermediate to Mid-Intermediate*	The moderate positive correlation with the Casual curve and the moderate negative correlation with the Intermediate curve are clearly not reflecting the experience level.
11	No Variation in the Experience-Level, thus no trend is obtainable.			*Lower-Advanced*	Not Applicable
12	0.58638	0.0748	-0.86803	*Lower-Advanced*	A strong negative correlation with the Advanced curve and a moderate positive correlation with the Casual curve. Once more clearly not reflecting the experience level.
13	-0.67977	0.55814	0.34507	*Upper-Intermediate to Lower-Advanced*	A moderate – respectively, negative and positive – correlations with both the Casual and Intermediate curves, accompanied by a weak correlation with Advanced curve.
14	0.00446	-0.66334	0.35246	*Mid-Intermediate*	The weak correlation with the Advanced curve and the strong negative correlation with the Intermediate curve are not reflecting the player's ranks.
15	-0.34864	-0.26389	0.37751	*Lower-Intermediate to Mid-Intermediate*	The advancement within the intermediate range is clearly not reflected by the weak negative correlation with the Intermediate curve.

Table 4.6 (Continued). The Correlation between the Selected Players' Ranks and the Competency Monitoring Curves *(Based on Connectivity-Patterns).*

4.4.5 Diagnosing the Learning Behaviour

The previous subsection, demonstrated using AC_P as an objective external-observer to classify and monitor the experience and learning behaviour of human Go players. This subsection is considered the final stage of these results, in which the learning activity of human Go-players is diagnosed by temporally observing each of the motifs' characteristics. The benefit of the following figures is the opportunity to realize how the connectivity-patterns of human Go-players is decomposed among the available motifs', and how those patterns evolve temporally with experience. Other benefits of such results, is the ability to construct customized learning processes that consist of designed tasks that convey to a learner the missing bits of skill *(which might, in this case, be inferred from the motifs counts)*, by which gaining experience might be assured and/or accelerated.

The figures below show the un-weighted Cumulative Moving Average (CMA) – again with a maximum window size of *50* games – of the Motif Cumulative Frequencies per Step (MCFS), as directly measured from the games. To allow mutual comparisons, each figure is normalized by the corresponding highest averaged characteristic-value. Also, for the sake of simplicity, the upcoming results will be shown only for the five players with the highest number of game-records. Those five selected players also cover the whole available experience range in somewhat mutually-exclusive sub-ranges.

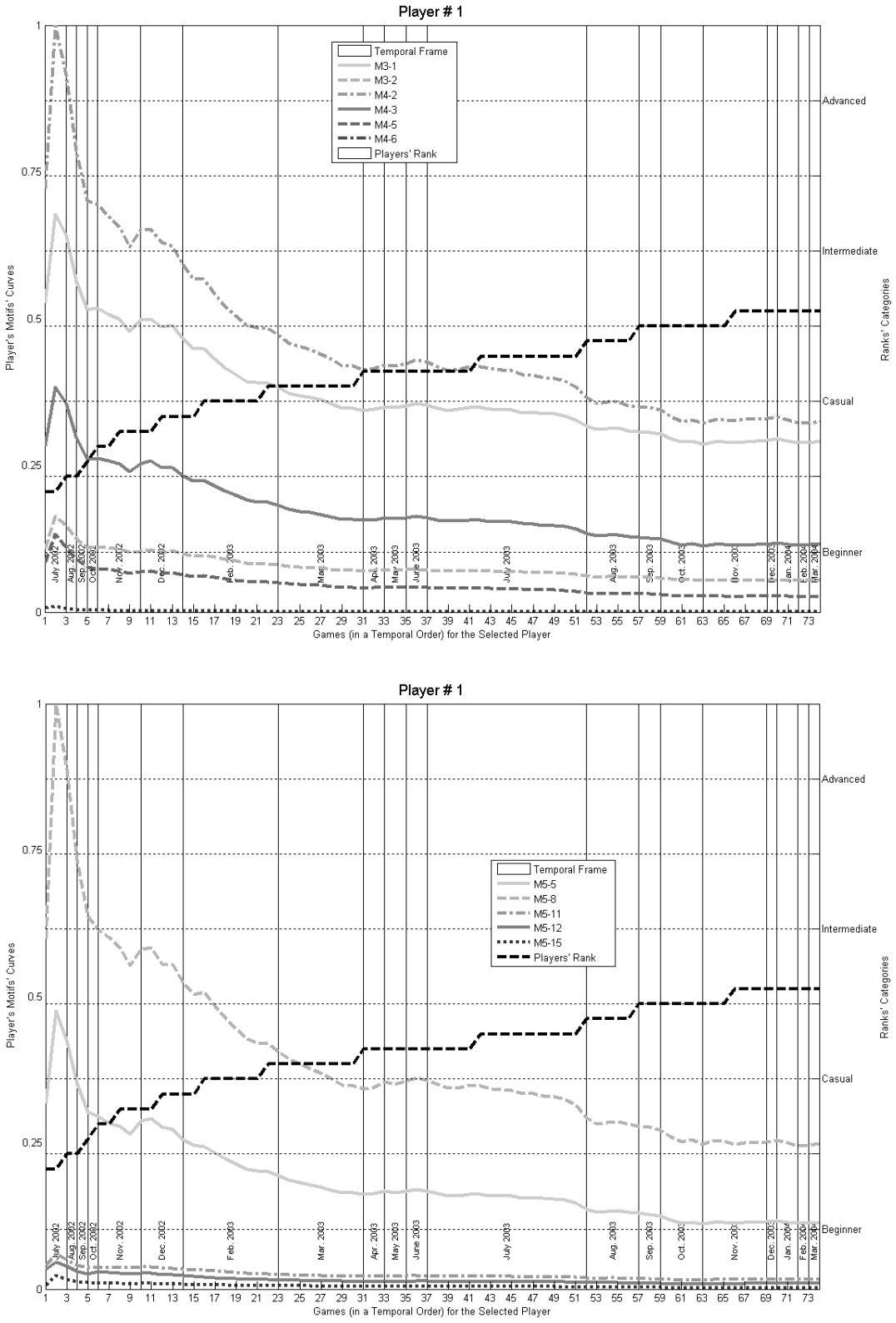


Figure 4.8 (a). The Skills – *Motifs Diagnosis* – Monitoring Curves – Player #1.

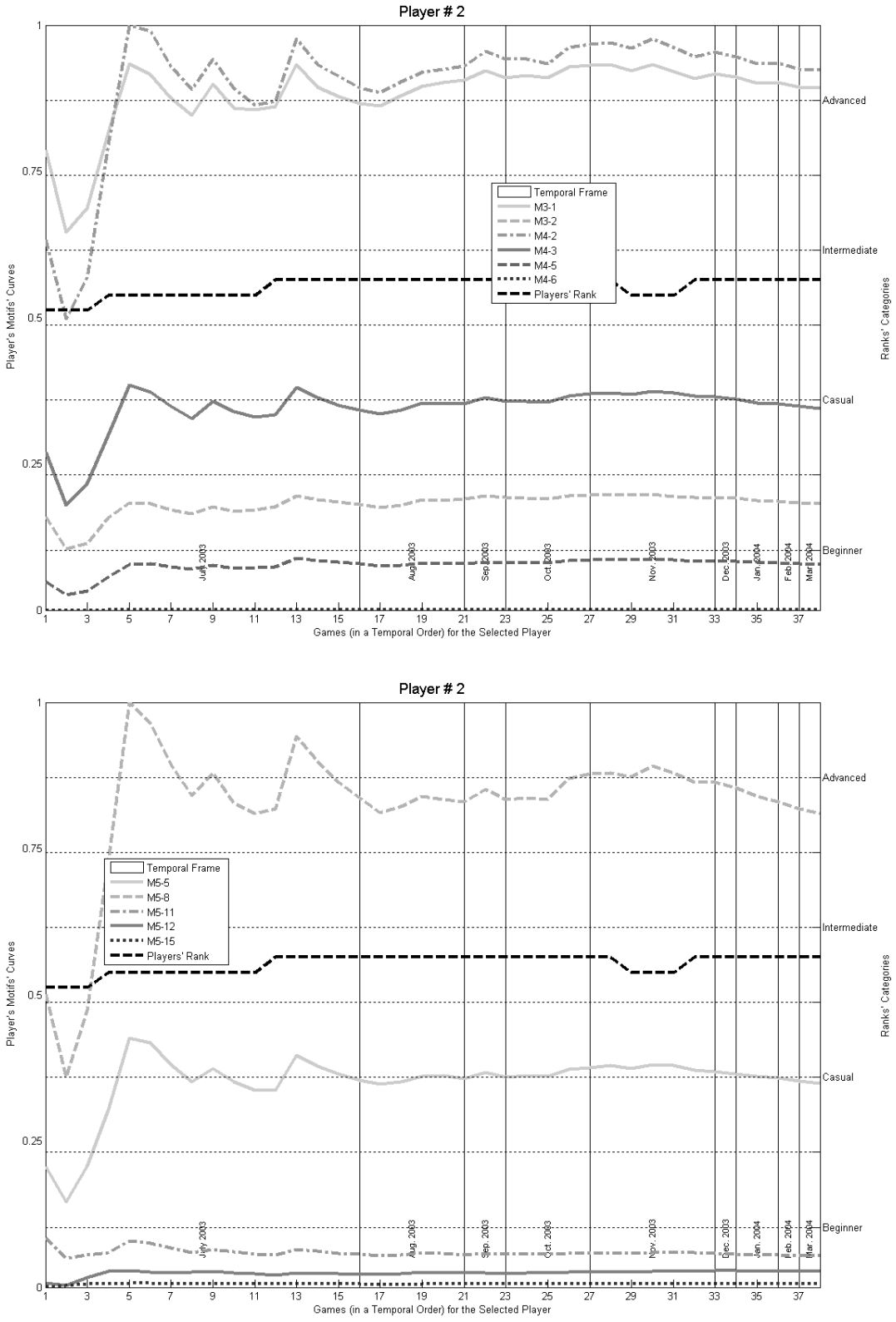


Figure 4.8 (b). The Skills – *Motifs Diagnosis* – Monitoring Curves – Player #2.

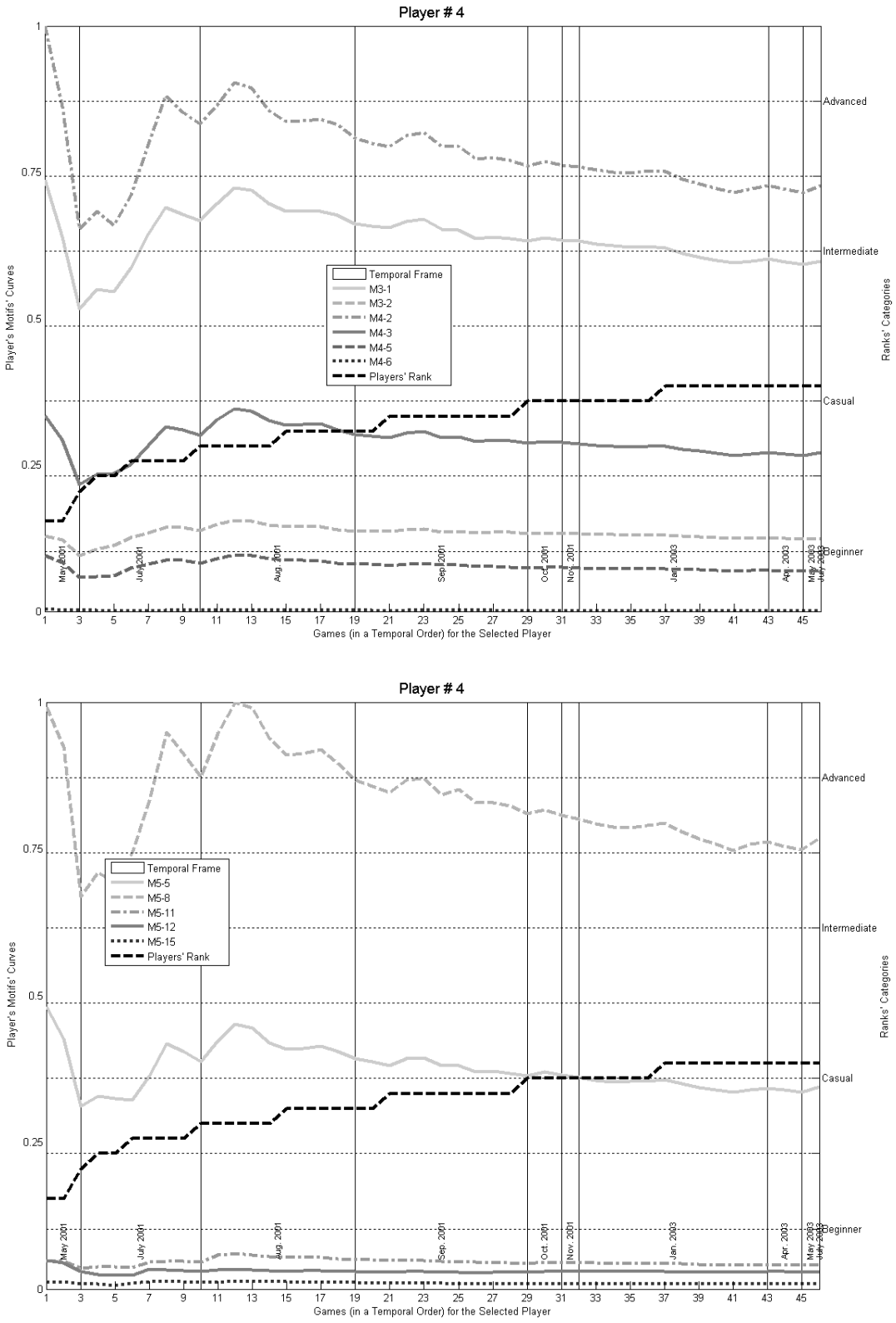


Figure 4.8 (c). The Skills – *Motifs Diagnosis* – Monitoring Curves – Player #4.

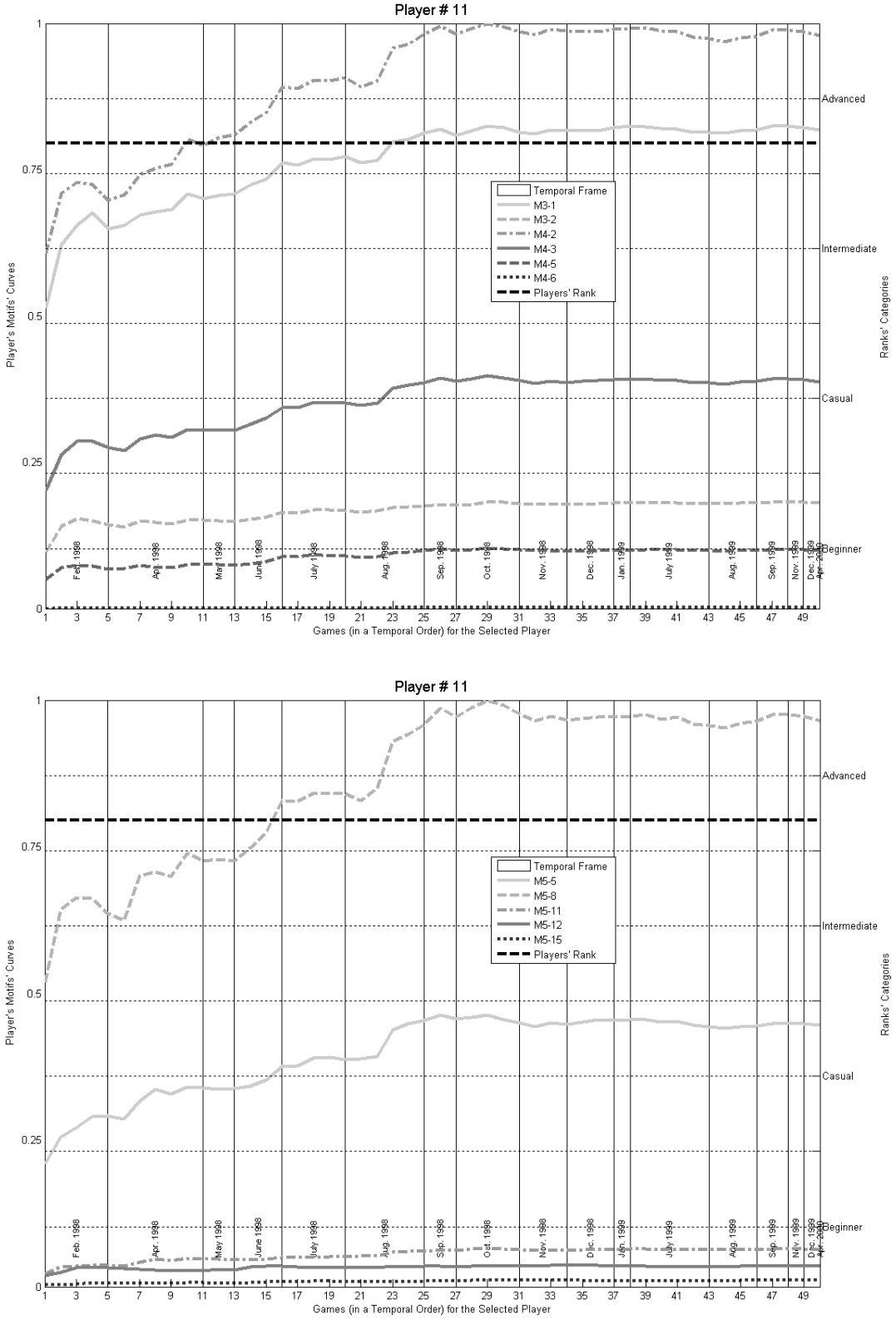


Figure 4.8 (d). The Skills – *Motifs Diagnosis* – Monitoring Curves – Player #11.

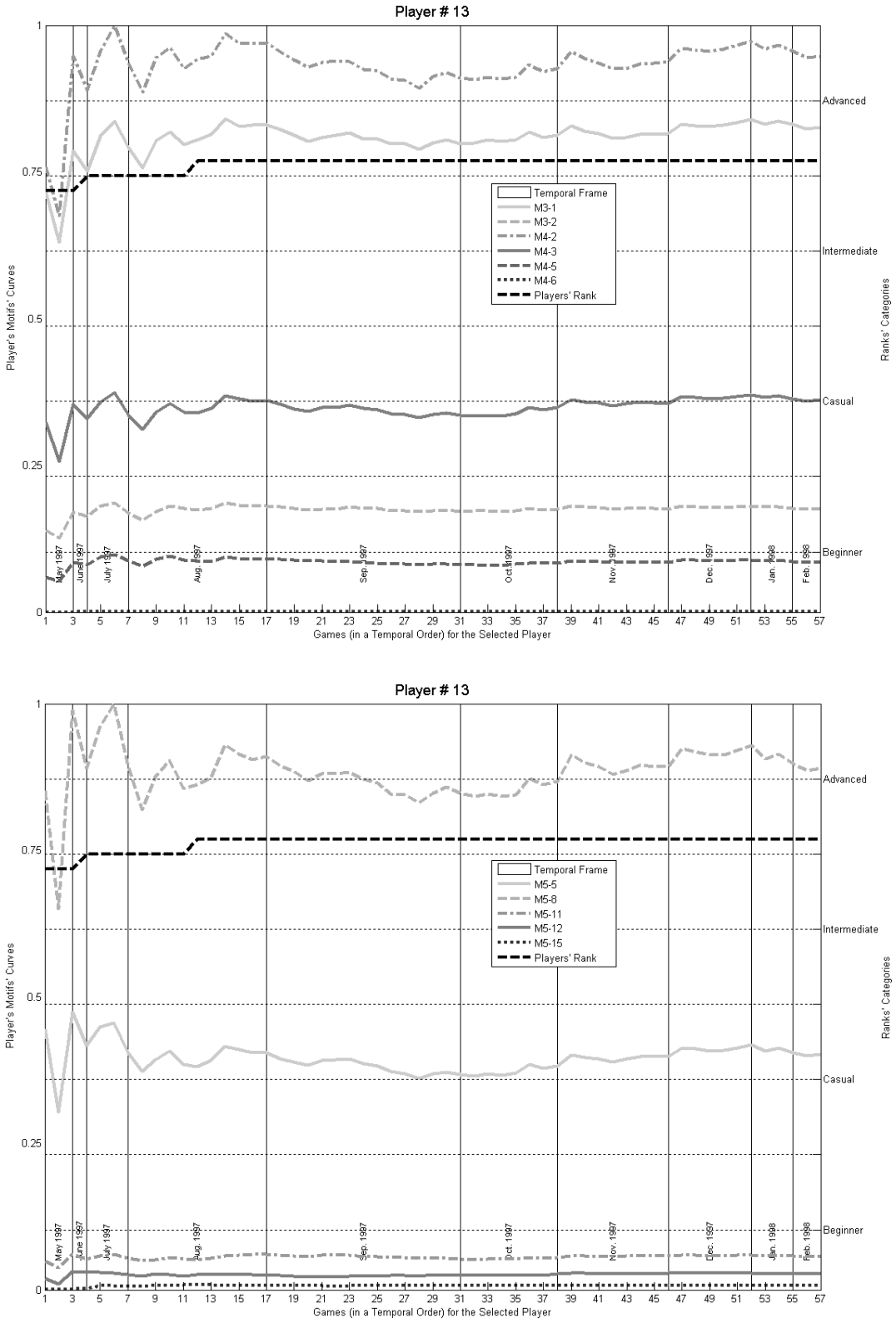


Figure 4.8 (e). The Skills – *Motifs Diagnosis* – Monitoring Curves – Player #13.

4.5 Chapter Summary

In this chapter, we have provided a methodology for an automatic and objective discovery of the properties of the human-players' connectivity-patterns employed in the game of Go, and to explore the changes in the patterns as the players gain experience.

In more details, we have presented a computational framework in which the cornerstone is a connectivity-patterns assessment component that provided an evaluation in terms of network motifs to the moves played by human Go players. Undirected motifs of three different sizes were investigated. The methodology successfully analysed a training dataset that is composed of hundreds of games that were suitably selected from an online server, and the results demonstrated a statistically-significant capability in characterizing varying levels of expertise in Go.

Based on the results of this analysis, statistical measures were applied to generate features upon which a competency model was developed. The model, developed as a three-tier ensemble-classifier based on random forests, was adequately capable of predicting the class label of a game of Go. This capability was demonstrated using a test dataset of *15* human players, whom which the proposed approach provided a temporal-monitoring for their overall expertise-level along with their specific skills-levels.

Chapter 5: Reasoning Competency in Human Go Players

When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind.
–––William Thomson, Lord Kelvin (1824-1907)

The work, reported in this chapter, has been partially published in following articles:

Amr S. Ghoneim, Daryl L. Essam, and Hussein A. Abbass (2011). Competency Awareness in Strategic Decision Making. In Proceedings of the IEEE First International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), pp. 106-109, 22-24 Feb. 2011, Miami Beach, FL, USA. IEEE Press.

Amr S. Ghoneim, Daryl L. Essam, and Hussein A. Abbass (2011). On Computations and Strategies for Real and Artificial Systems. In Advances in Artificial Life, ECAL 2011: Proceedings of the Eleventh European Conference on the Synthesis and Simulation of Living Systems, edited by Tom Lenaerts, Mario Giacobini, Hugues Bersini, Paul Bourgine, Marco Dorigo and René Doursat, pp. 260-267, 8-12 August 2011, Paris, France. MIT Press.

In this chapter, we introduce an adaptation to the novel approach proposed in Chapter 4. The aim remains that of quantifying and monitoring the competency of human Go players, yet in terms of reasoning competency. The chapter starts in Section 5.1 with an overview of the problem to be tackled and its formal definition. In Section 5.2, the proposed adaptation – *of the previously proposed methodology* – to attempt this problem is presented, including a range of reasons to be assessed. Section 5.3 investigates the potential of the assessed reasons to capture and reflect the competency of Go players using the formerly selected training set and simple statistical methods. Once more, the statistical methods include measures of dispersion to characterize the changes in reasoning among the selected levels of expertise, and are followed by tests of significance on whether the calculated measures can reliably distinguish between those with varying levels of expertise. In Section 5.4, and as beforehand, the new calculated

measures are used as features to construct the proposed classifier, which is then utilized to temporally monitor the competency of the cases in the testing dataset. The section concludes by a discussion of the achieved results, and by relating the reported performance and results to those of Chapter 4. Section 5.4 also includes additional investigations of the effects of changing particular experimental settings on the reported results. As a final point, the contributions in this chapter are summarized in Section 5.5.

5.1 Overview

As has been determined previously in Chapter 3, and as attempted in Chapter 4, the objective is to introduce a computational framework that automatically assesses the competency of Go players; *so once more, to attempt to answer the research sub-questions 1 and 2 (see Section 1.3)*. On the other hand, this chapter intends to address those sub-questions using situational reasoning, instead of connectivity-patterns. The objectives of the automation remain the same; that is, to provide a relatively detailed assessment when compared to the current automated approaches, while simultaneously reducing the need for assessments performed by human experts, which – *in spite of their in-depth nature* – are invasive, subjective, and qualitative.

5.1.1 Related Work

Situation reasoning, within the generic literature, is addressed using varying approaches and for numerous applications; by the standard ontology language OWL for mobile communication (Luther et al., 2005), by first-order logic for pervasive computing (Yau and Liu, 2006), and by time-series of sound information for situation recognition in humanoid robots (Tokutsu et al., 2009). Within the Go domain, reasoning – *which is considered as a knowledge-based approach* – depended mostly on patterns, which combine the basic concepts of the game *(board intersection, stone, neighbourhood, liberty, connection, eye, and group)* and thus constitutes the fundamental level on which any further spatial reasoning is based (Bouzy, 1996; Bouzy and Cazenave, 2001; Müller, 2002). Accordingly, a higher level of spatial reasoning is built iteratively, using additional patterns *(connectors, dividers … etc.)*, morphological

operators, or propositional logic. Nonetheless, reasoning in Go has been far from trivial, and even attempts to automatically generate Go rules – *in terms of patterns and particular associated conditions* – for improving tactical reasoning have required high levels of computational effort for all of generation, storage, and utilization (Cazenave, 2002). Section 4.2 provides details on the main categories of reasoning in Go, followed by the proposed adaptations to the assessment approach.

5.1.2 Problem Statement

The problem to be attempted is: To automatically, passively, quantitatively, and objectively assess the competency of Go players by characterizing the decision-making performance of human Go players in terms of explanatory in-depth situation reasoning that links a particular board situation to the consequent executed action (i.e. move). The reasoning is determined using a human-inspired procedure to the Go game; that is, given a board situation, how a player perceives the board and compiles its basic structures, and how he/she explores, investigates, analyses the moves and plans in order to decide upon a final action. As presented in Chapter 4, the procedure is performed using a fully computerized simulation, and is used to analyse a large cohort of games played by – novice, amateur, and skilled – human Go players. In other words, the results expected to be achieved – *corresponding to those defined in Chapter 4* – are:

- That given a set of Go *"Games"* performed by human players of varying *"Expertise"* levels, the difference in the expertise can be explained by a statistically-significant quantitatively-based difference in the sets of in-depth reasoning *"$Reasoning_{Expertise}$"* estimated from the corresponding *"$Games_{Expertise}$"* using the proposed reasoning assessment.

- And consequently, that the quantitative measurements representing a specific *"Expertise"* level can be utilized to create a corresponding *"Competency Model"*. So that, given a set of Go *"Games"* played by a particular *"Player"* over a period of time, the *"Competency Model"* will reflect and monitor the changes in the performance of that specific *"Player"*.

5.2 Methodology

To accomplish this, we present an adaptation (Figure 5.1) to the computational framework previously shown in Figure 4.1. The cornerstone of this framework – *instead of the pattern assessment component (AC_P)* – is a reasoning assessment component (AC_R), which can be thought of as an *external observer* who provides an explanation – in terms of declarative reasons – to the moves played by the Go players. The AC_R relies on specific definitions of the selected standard concepts used in the Go domain to justify the selected moves. As a result of this fully automated approach, and in line with the former AC_P, the AC_R can act as the *reference point* by which the reasoning competency of different players can be compared, assuring the objectiveness, passivity, and non-intrusive nature of the provided analysis.

In order to investigate the usefulness of the estimated reasons in expressing the competency of Go players, and thus in discriminating between varying performance levels, we include – *as beforehand in Section 4.2* – a preliminary phase that attempts to characterise those varying levels in terms of the estimated reasoning. This phase, corresponding to steps from 1 to 3 in Figure 5.1, includes: 1) reanalysing the *training dataset* (selected to reflect the different available levels of performance in Go) using the AC_R, 2) calculating the statistical dispersion measures – *the Median, and the Median Absolute Deviation*, and finally 3) applying tests of Statistical Significance to demonstrate the effectiveness of the generated reasoning as features capable of describing and – accordingly – distinguishing the varying levels of performance in Go. Prior to applying the tests, a Multivariate Statistic will be applied to the features to detect the presence of any outliers.

Once this phase is realized, the features are then used to construct the ensemble-based classifier *(proposed in Figure 4.5)* capable of learning the different existing patterns of reasoning. The construction of the classifier begins the second phase of our proposed methodology – steps from 4 to 6 in Figure 5.1 – which includes the monitoring of the competencies as the players' experiences vary with time. Temporally-ordered games are compiled for each case, as in the previously selected *testing dataset*,

and are analysed by the AC_R to generate the set of features to be classified. Finally, the votes from the trained classifier – *a vote for each game* – are plotted to show how the learning process is evolving, and to estimate the current overall competency/experience level for each selected player. A temporal plot of these features can be used to present a more detailed skill assessment for the selected players. The remaining sections in the chapter describe – consecutively; as shown in Figure 5.1 – each step in the framework. Coming next are the definitions of the main concepts of reasoning in the game of Go.

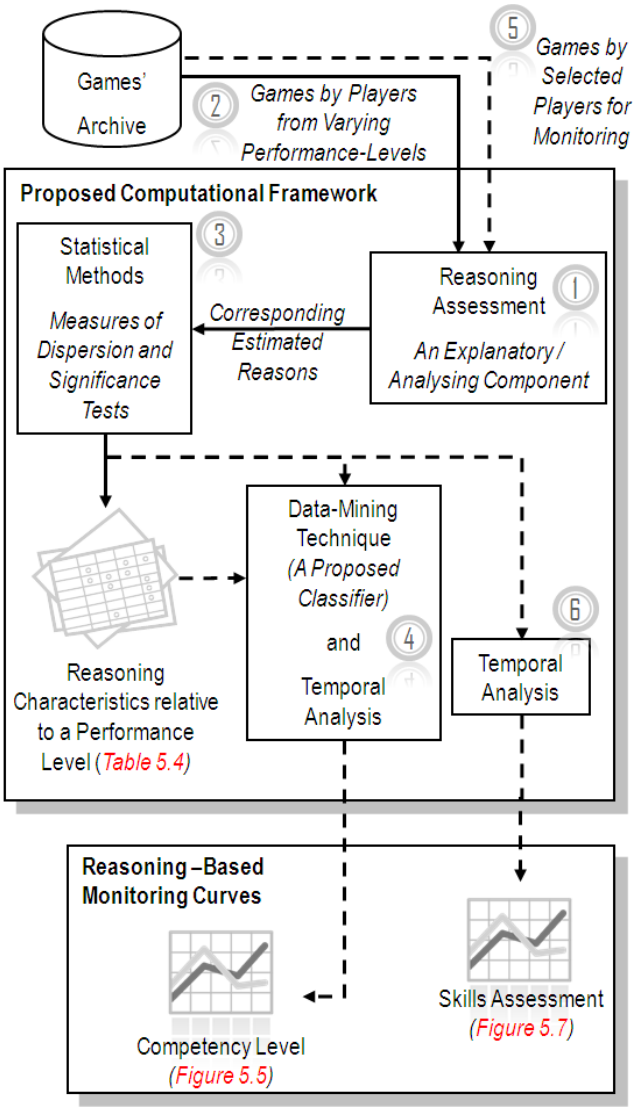


Figure 5.1. The Proposed Adaptation to the Computational Framework formerly proposed in (Figure 4.1).

5.2.1 Concepts of Reasoning in Go

The reasoning used to justify decisions in a game of Go considerably varies throughout the game, and undoubtedly depends on the current board situation. Yet, the reasoning can be captured in general useful concepts. Those concepts – *which we have previously defined in Section 2.1* – are identified by observing human players and can be explicated by Go experts. The concepts can be defined by considering several factors; the current stage of the game, the level of look-ahead required, the required effect, and the scope – *of the board* – under consideration. The stage of the game leads to the concepts of *End-of-the-Game* and *Opening*, both of which are usually generated using databases of recommended patterns. The fact that numerous lines of play are recommended – *through experience* – gives rise to the concept of *Anti-Suji*; that is, unskilled moves. The majority of the moves in a game deal with creating and securing groups of stones, either by:

- *Attacking* the opponent's groups by, for instance, a *Cutting* move that prevents opposing stones/groups from being connected.
- *Defending* one's groups by, for instance, a *Connecting* move that maintains one's stones/groups connectivity.
- *Reading* or *looking-ahead* for determining in advance whether a specific group is indefinitely alive – and thus going to remain on the board – or dead – and thus will eventually be captured.
- Immediate *Capturing* of opposing stones through tactical – *i.e., local* – moves, or of *Invading* the opponents territory by strategic – *i.e., more global* – moves that will expand one's Territory and/or Influence.

For an additional demonstration of those concepts, the following set of figures will illustrate through a famous Go game – *known as the "Dosaku's Masterpiece"* – the reasoning behind selected moves. Honinbo Dosaku (1645–1702) was a professional Japanese Go player who is regarded by many as the greatest player of all time. Dosaku

played his masterpiece – *on the 19th of November 1683* – as white, giving his opponent – *Yasui Shunchi; one of the strongest Go players* – 2 handicap stones, and yet managing to lose the game by only 1 point. The reader is reminded that a definition for each of the Go terms used in this section can be found in Section 2.1.

Figure 5.2 (a)–(h). The Dosaku's Masterpiece *(the 4 upcoming pages)*:

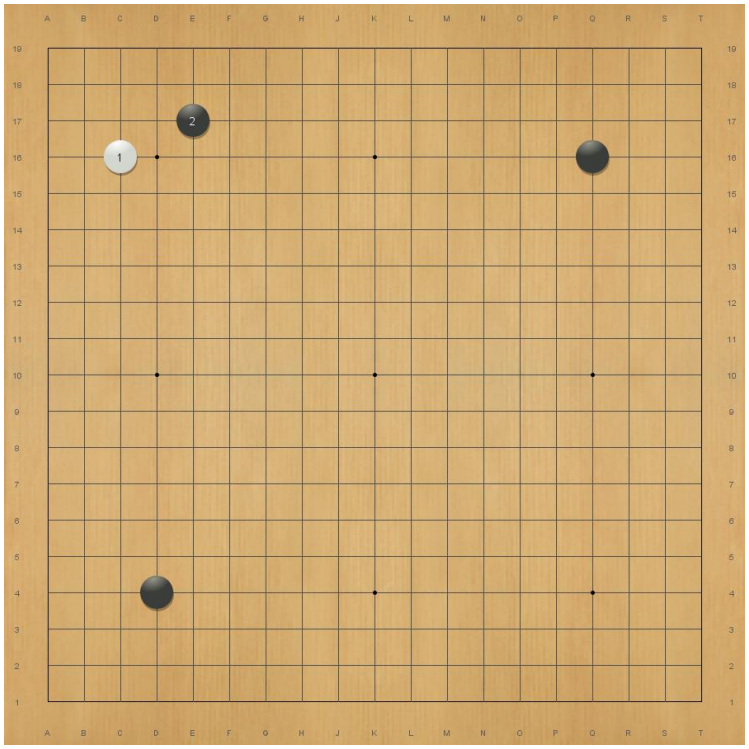
(a) The Black move at E17 – *numbered 2* – Fuseki recommendation, expands Black's territory and Moyo, and is considered a strategic attack on C16, ***(b)*** White answers by the move O17 – *numbered 3* – in turn expands White's territory and Moyo while strategically attacking Q16.

(c) White move at N4 – *numbered 31* – connecting and defending both N3 and M4 while cutting and attacking M3 and M5, ***(d)*** Black responds by a move at L3 – *numbered 32* – that defends M3 and connects it to L4.

(e) The Black move at F2 – *move number 60* – captured E2 and thus connected and defended all the surrounding stones – *D2, E1, and E3* – in addition to defending D4 and strategically defended the group centred at C6. Consequently, the move severed the connections between the 3 surrounding white groups – *(C3, D3), (D5, D6), and (F3, F4)* – and therefore is considered a strategic attack on all of the three groups in addition to G2. ***(f)*** The White move at P12 – *numbered 253* – defends the group while strengthening the connection towards both the groups including Q11 and the group containing O15. The move threatens O11 and cuts it from P13 and its surroundings.

(g) The Black move at A13 – *numbered 260* – attacks and captures A14, thus it expands Black's territory, and defends the group (A15, A16) by connecting it to the group centred at B14. ***(h)*** The White move at M16 – *numbered 273* – successfully attacks and captures the group (M17, N17) and thus defends the group by supporting its connections to the surrounding White groups *(the group that includes L17 and the group that includes M18)*. The move thus strategically threatens many of the surrounding Black stones including O16 which will be captured 4 moves later.

(a)



(b)

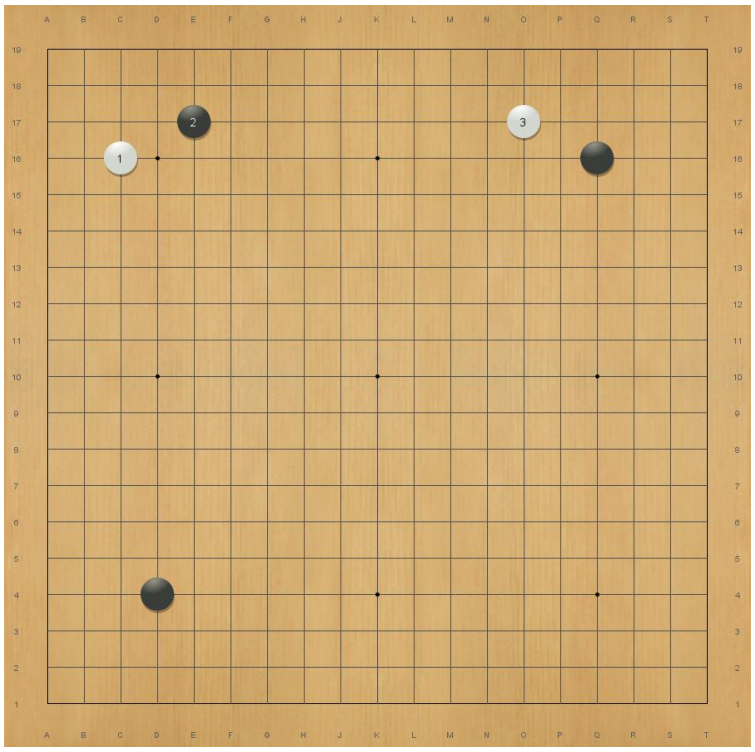
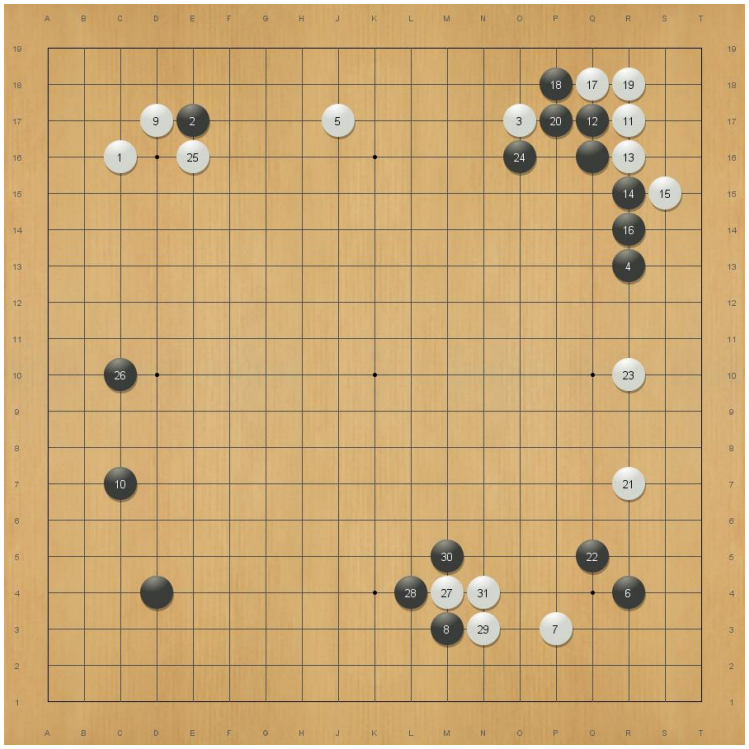


Figure 5.2 (a)–(b). The Dosaku's Masterpiece: (a) Black moves at E17 – *numbered 2*, (b) White answers by the move O17 – *numbered 3*.

(c)



(d)

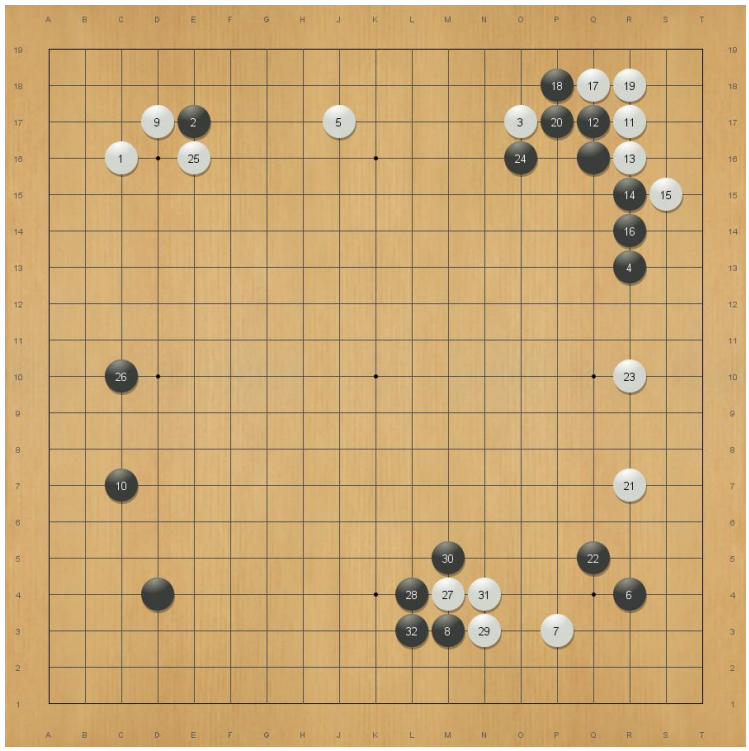
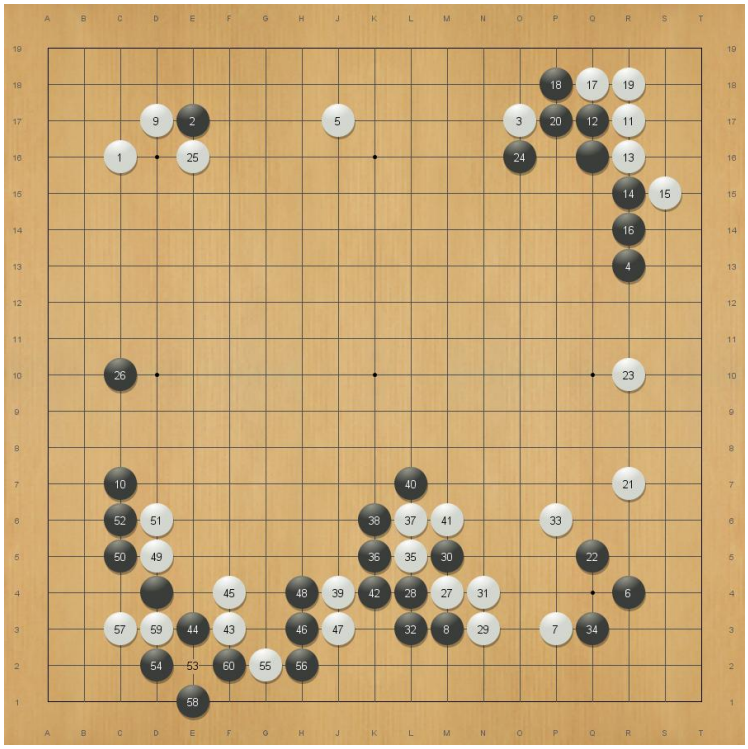


Figure 5.2 (c)–(d). The Dosaku's Masterpiece: (c) White moves at N4 – *numbered 31*, (d) Black responds by a move at L3 – *numbered 32*.

(e)



(f)

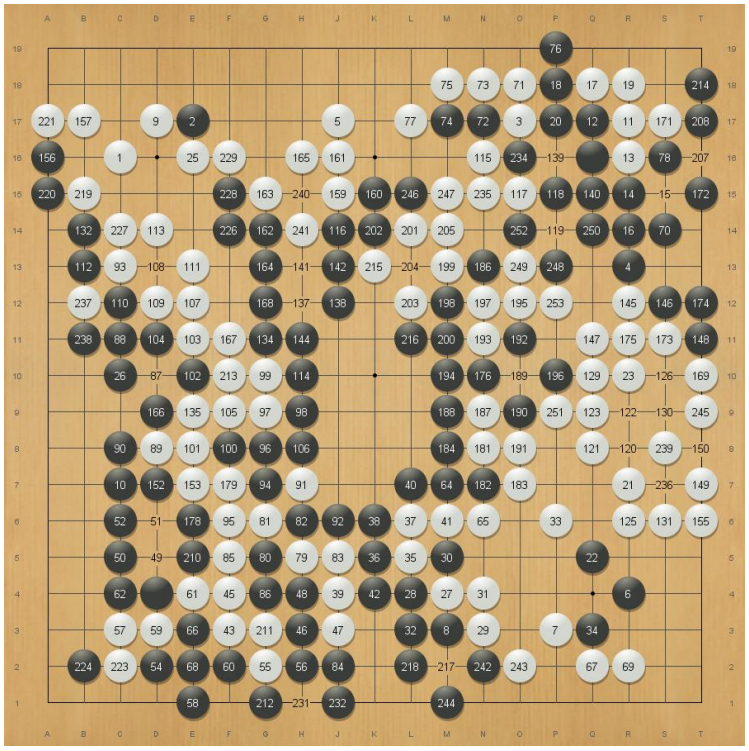
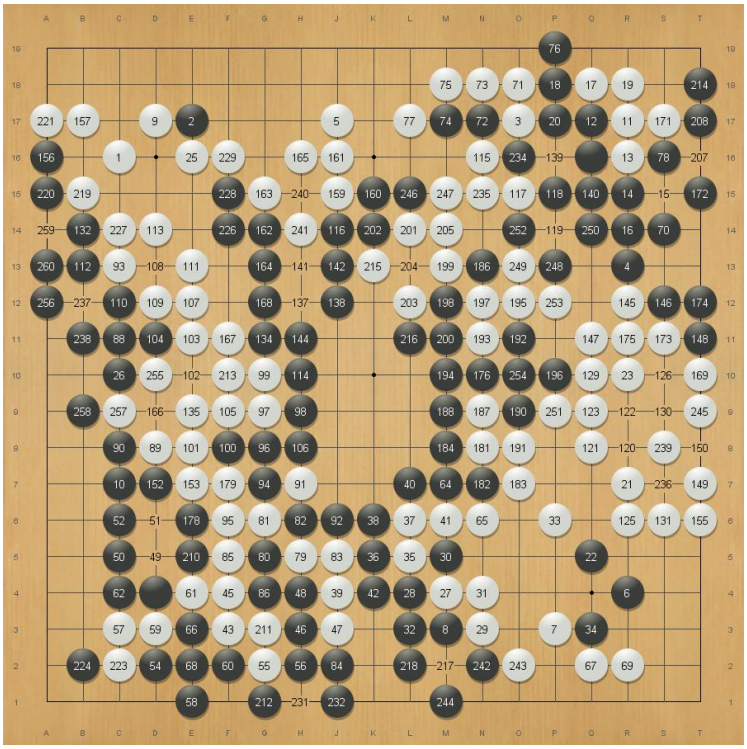


Figure 5.2 (e)–(f). The Dosaku's Masterpiece: (e) The Black moves at F2 – *move number 60*, (f) The White moves at P12 – *numbered 253*.

(g)



(h)

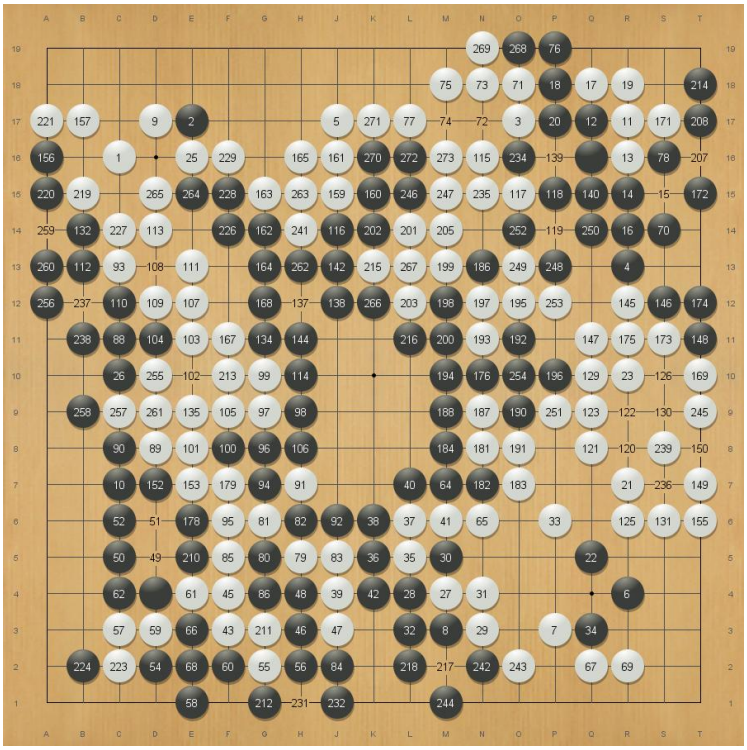


Figure 5.2 (g)–(h). The Dosaku's Masterpiece: (g) The Black moves at A13 – *numbered 260*, (h) The White moves at M16 – *numbered 273*.

5.2.2 A Publicly-Available Reasoning Assessment Method

In order to generate the Go reasoning, we propose extending the roles of the available engines (Computer-Go programs) to sub-serve in the analysis by acting as the AC_R. That is, some of the available engines – *whether commercial or publicly available* – that model – *with varying degrees* – the functionality of Go players can be used as an alternative to constructing new models. The reasons for that are, first, a large investment has been put into many of those engines; financially and/or in terms of man-hours, but also, most importantly, a significant amount of patterns and/or rules based on Go masters/experts have been incorporated into them. Second, engines are primarily developed to play on online servers, therefore they (i.e., the engines) can be effectively used to set up any experiments that require interaction with other humans or Go engines. Finally, the tuning parameters available in most engines simplify any required tuning/adding to the integrated methods, rules, or datasets.

GNU Go is one of the leading non-commercial computer-Go programs, and plays regularly on online Go servers and computer-Go tournaments (D. Bump et al., 2009). The GNU Go program uses a multi-level approach for choosing its moves (Figure 5.3), culminating with a group of hand-tuned heuristic rules that combine a set of characteristic values and weights. That set of twelve characteristic values represent the program's evaluation of the generated move reasons, while the weights are used to carefully combine those characteristic values depending on the style of the current move *(a combination of the current score and the game status).* The reasons for a move are assigned using the move generator, which relies primarily on a range of pattern matchers to decide which tactics each investigated move achieves. The move generator takes into consideration an estimated influence calculated for both players. Finally, all these modules rely on an internal representation of the board that must be initially compiled, and again this is partially dependent on the pattern matchers. The move generator may incorporate the output of a Monte Carlo module, which plays random games to the end, and that also generates its moves from a pattern database. The GNU Go engine comes with development and debugging options, such as *–trace* which

displays more detailed debugging information. Combining that with GNU Go's ability to replay a game, we can estimate at each step (move) of the game what were the possible reasons behind it.

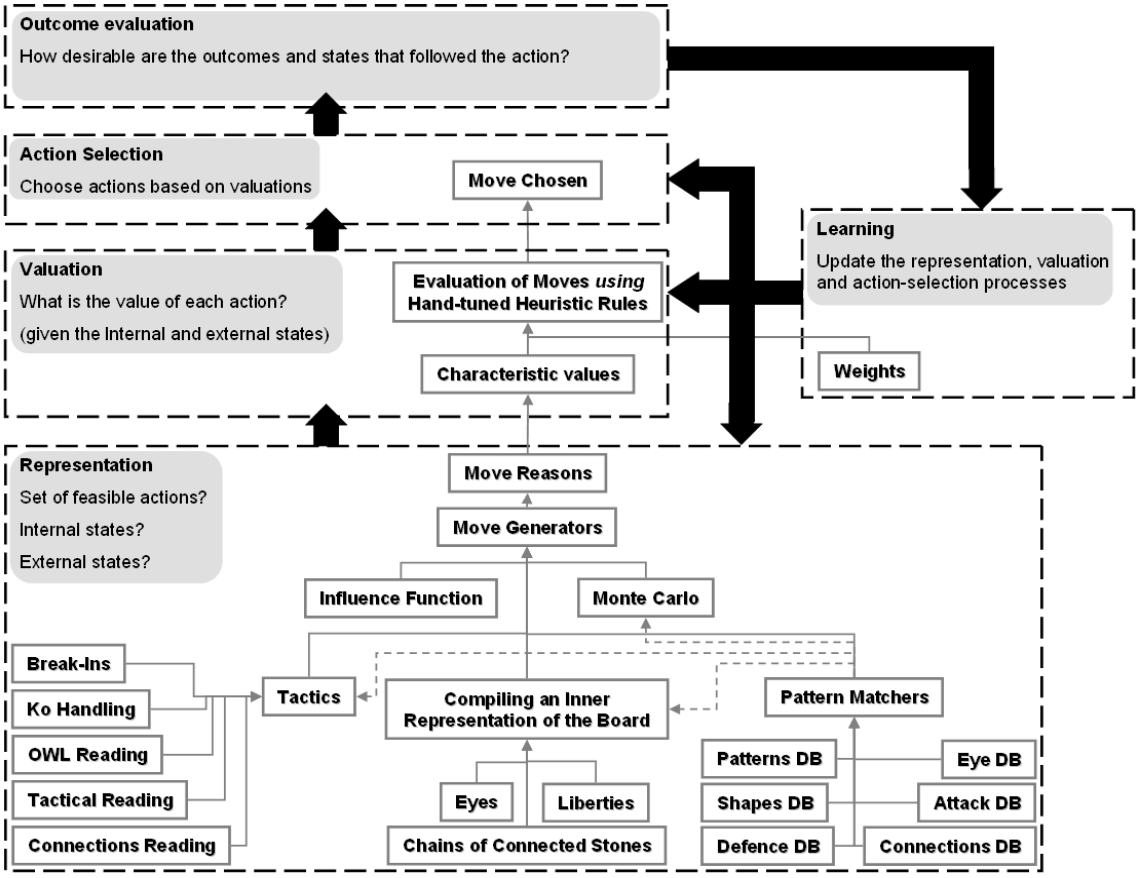


Figure 5.3. Basic Computations involved in making choice (light gray) and the corresponding GNU Go computational modules.

GNU Go, especially in its way of compiling the board representation and generating strategic reasons, can be following to an extent by a more human-inspired reasoning model, especially when compared to other available Go engines. Whether GNU Go is adopting a human-inspired model or not, and the extent of this 'adoption' will not be discussed here, it is appropriate to limit this adoption – if existing – to a semantic or knowledge level. In addition, it follows in its design one of the prevalent models of decision making, namely that of *value-based decision-making* (Rangel et al.,

2008), which occurs whenever a decision is taken based on subjective-values placed upon the alternatives. Based on the various theoretical-models of value-based decision-making that existed in several disciplines *(i.e., computer science, economics, and psychology)*, Rangel et al. (2008) proposed the five-basic processes that were assumed—by the majority of models in the previously mentioned disciplines—to be performed whenever a value-based decision is made. Figure 5.3 shows the proposed five *"basic computations involved in making choice"* (Rangel et al., 2008) combined with the 'virtually' corresponding GNU Go computational modules.

Our main aim is to discover the key properties of the strategies employed. Therefore, grouping the reasons into a small number of sets according to their tactical/strategical similarities will notably assist in identifying the general strategies and observing any potential trends regarding the evolvement of those strategies with experience. Therefore, R can be defined as $R = s_1 \cup s_2 \cup \ldots \cup s_n$, where s_n is the *nth* subset. The aggregated sets of reasons proposed for this design are:

1. Not recommended; includes reasons for moves that are deemed as unsafe from a strategical perspective. Anti-suji[3] moves are included in this.

2. Considered an Attack; includes all the moves that tactically capture a chain[4] or a group[5]. The set generally consists of attacks, attack threats, multiple[6] attacks, and moves which prevent a connection between two chains.

3. Considered a Defence; includes reasons for all the moves that defend a chain or a group from being attacked. The set generally consists of defence moves, threats to defend, multiple defends, and moves which connect two chains.

[3] Anti-suji are forbidden, crude, and/or locally-inferior moves.

[4] Chains are vertically and horizontally adjacent stones of the same colour.

[5] A Group is a set of amalgamated chains which are treated as a unit.

[6] Moves that attack/defend more than a single chain simultaneously.

4. Explicit Gains; includes reasons for tangible achievements, such as moves that directly capture something, invade the opponent's territory, or increase one's own territory and/or region of influence.

5. Thoughtful; includes reasons dealing with *life and death problems*[7]. In GNU Go, these types of reasons are implemented by the "Optics With Limit-negotiation" (OWL) module which visualizes potential moves to determine the result of various moves.

6. End of the Game; includes reasons for the typically small moves that occur late in the game. By approaching a reasonably stable territory, these moves typically expand one's own territory while reducing the opponent's.

7. All Reasons; is the universal set *R* that contains all conceivable reasons. However, for simplification, this set contains all the reasons that appeared at least once in the selected dataset, rather than all the reasons that might possibly be generated from the GNU Go engine.

In a preliminary step, we found thirty-forty distinct reasons *R* that were generated by the GNU Go for all the moves in our selected dataset. The available reasons are categorized into the seven "overlapping" sets previously mentioned *(See Table 5.1 for more details)*. It is worth mentioning that several of the reasons consider more than a single move forward. Thus, a short-term-based evaluation will not be produced by relying on the generated reasoning to analyse a Go game.

After explaining the concepts of reasoning in Go and the reasoning assessment method – *which constitute the reasoning assessment component (AC_R)*, Figure 5.4 describes the remaining processes of the proposed computational framework in relation to the basic steps of data-mining. Those remaining processes are detailed in the following Sections 5.3 and 5.4.

[7] A fundamental concept in the game of Go that deals with whether the status of a distinct group is alive or dead, and requires deeper analysis in comparison to other problems.

Table 5.1. The Available Reasons and Their Corresponding Sets.

	The Reasons Available	**Reasons' Sets**						
		Not Recommended	**Considered An Attack**	**Considered A Defence**	**Explicit Gains**	**Thoughtful**	**End of the Game**	**All Reasons**
1	threatens to attack *X*		■					■
2	strategically or tactically unsafe	■						■
3	expands territory				■			■
4	strategically defends *X*			■				■
5	defends *X*			■		■		■
6	owl-defends *X*			■				■
7	connects *X* and *X*							■
8	attacks *X*		■					■
9	owl-attacks *X*		■			■		■
10	cuts *X* and *X*		■					■
11	threatens to defend *X*			■				■
12	backfilling dame found for string *X*						■	■
13	strategically attacks *X*		■					■
14	expands moyo				■			■
15	Fuseki player suggestion							■
16	defends against combination attack			■				■
17	wins semeai for *X*							■
18	both defends *X* and defends *X*			■				■
19	Endgame move						■	■
20	is an invasion				■			■
21	either attacks *X* or attacks *X*		■					■
22	captures something				■			■
23	is a blunder, subtracting *X*	■						■
24	attacks *X* with bad ko		■					■
25	defends *X* with bad ko			■				■
26	owl-attacks *X* with bad ko		■			■		■
27	defends *X* with good ko			■				■
28	owl-defends *X* with good ko			■		■		■
29	attacks *X* with good ko		■					■
30	owl-attacks *X* with good ko		■			■		■
31	owl-defends *X* with bad ko			■		■		■
32	is an antisuji	■						■
33	attacks *X* (defenceless)		■					■
34	attacks *X* (defenceless) with good ko		■					■

A light grey cell indicates the corresponding reason's membership of one or more of the available sets.

***X* represents an intersection point on the Go board *(typically, a 19×19 Grid)*.**

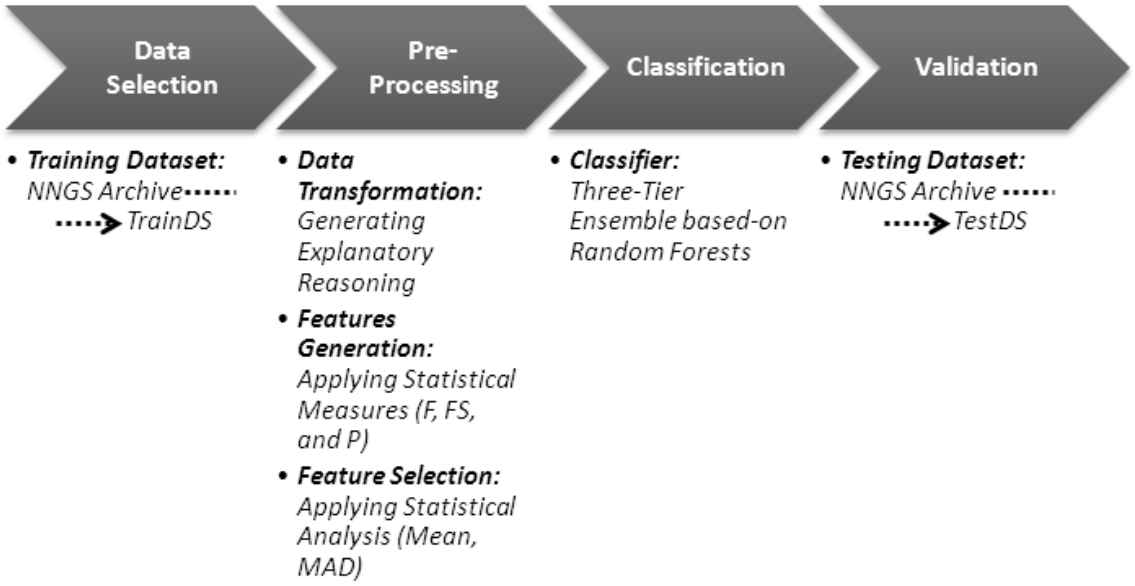


Figure 5.4. An overall illustration of the basic processes for assessing the competency of Go players based on Situational Reasoning; detailed in Sections 5.3 and 5.4.

5.3 Estimation of Competency Models

This section presents the first phase of the methodology, in which we ran our system using the *Training Dataset*, which consists of games played by casual, *intermediate*, and *advanced* human opponents. The AC_R will replay the games and generate reasons for the majority of the moves, thus estimating the strategical reasoning originally employed by its human players. An initial pre-processing step is then carried out by applying 1) frequencies measurements and 2) an outliers' detection algorithm. The frequencies measurements – which includes measuring the Frequencies *'F'*, Frequencies per Step *'FS'*, and the Percentages *'P'* – of the aggregated subsets of the generated-reasons per game create the final set of measures that are generated for each game. This step is followed by the final statistical analysis in which measures of dispersion and tests of significance are used to examine the implications of the final features sets.

5.3.1 Training Dataset

The dataset used for training in this chapter is *trainDS*, as previously discussed in Section 4.3.1, and based on the traditional *kyu/dan* ranking system of Go. The *381* games were selected from the NNGS online server, and are played by human Go players on the standard *19×19* boards. The selected games represent three levels of performance – *Casual, Intermediate, and Advanced* – each with *127* games.

5.3.2 Data Pre-processing

The GNU Go move-generator lists – *if available* – a varying number of possible reasons for each move *(occasionally, a move may have no reason from the engine's perspective)*. The main features then calculated from the reasons list per game are the three following measurements:

- Reasons Frequencies *(F)*; the frequency of each reason per game. Given the set of all reasons *R*, *F* is defined as:

$$F_{ij} = \sum_{r=1}^{N_{rows}} \sum_{c=1}^{N_{cols}} IP(r,c,R_j) \qquad i = 1 \ldots N_{games},\ j = 1 \ldots N_{reasons} \qquad \text{(Eq. 5.1)}$$

  where the function *IP* returns 1 iff the reason R_j was generated for a move played at the Go-board's Intersection Point IP_{rc}, otherwise, *IP* returns zero. If the reason R_j was generated separately for more than a single move at IP_{rc}, *IP* returns 1 for each such move. N_{games} and N_{reaons} are respectively the total number of games and reasons available, while N_{rows} and N_{cols} are the number of rows and columns in a Go board.

- Reasons Frequencies per Step *(FS)*; the frequency of each reason per game divided by the total number of steps *(i.e. moves)* in the corresponding game, defined as:

$$FS_{ij} = F_{ij} \big/ N_{steps}(i) \qquad \text{(Eq. 5.2)}$$

where $N_{steps}(i)$ is the total number of moves in game i.

- Reasons Percentages *(P)*; the percentage of each reason *(from the total number of reasons generated)* per game, defined as:

$$P_{ij} = (F_{ij} * 100) \Big/ \sum_{k=1}^{N_{reasons}} F_{ik} \qquad \text{(Eq. 5.3)}$$

Prior to our statistical analysis, we detected outlying observations; that is, games in our dataset *"which appear to be inconsistent with the remainder of that set of data"* (Hawkins, 1980) therefore producing *"suspicion that [those games were] generated by a different mechanism"* (Johnson, 1992). In this study, outliers are not considered noise or error, rather they are assumed to carry important information that accounts particularly for any unaccounted for parameters when selecting the dataset (for example, the length—number of moves/steps—per game). Due to the multivariate nature of our observations (i.e., per game, different sets of reasons are estimated using 'potentially' various measurements), a multivariate statistic to determine the outliers—if any—is preferred over an univariate in order to decrease the effect of masking and/or swamping (Davies & Gather, 1993).

The Minimum Covariance Determinant *(MCD)* is a multivariate estimator that may be simply defined as one that finds an optimal subset of h observations—from the total n observations—with the smallest covariance-matrix determinant, with $n/2 < h < n$. The MCD then estimates the location and scatter of the data as respectively the arithmetic mean and covariance matrix of this optimal subset. The MCD was introduced by Rousseeuw (1985) as a robust estimate that is not excessively influenced by outliers, even with the presence of many outliers in the data.

This robustness can be given in terms of the Breakdown Point[8] *(BDP)* which is related—in the case of MCD—to the size of the optimal subset h; the typical $h \approx n/2$

[8] The Breakdown Point (BDP) is defined as *"the minimal fraction of contamination which renders the estimation meaningless"* (Fauconnier & Haesbroeck, 2009).

corresponds to the maximum BDP value of *50%* (Fauconnier & Haesbroeck, 2009). The size of the optimal subset h is usually to an input, or rather it is determined using the input parameter α, where $\alpha \in [1/2, 1]$. Thus, given the default $\alpha = 0.5$, h can be approximated as $h \approx \alpha * n$, and the $BDP = 1 - \alpha$.

Nevertheless, this default value for α ensures the highest resistance to outliers by means of inefficiencies indicated by Type I errors; non-outliers tagged as outliers. Therefore, Fauconnier and Haesbroeck (2009) showed that a BDP of *25%* (i.e., $\alpha = 0.75$) is a practical solution that balances robustness and efficiency. A fast-MCD algorithm with several time-saving techniques was proposed by Rousseeuw and Van Driessen (1999) and is used throughout this study.

As we have just mentioned, the value of $\alpha = 0.75$ can be considered a practical solution that maintains both the robustness and effectiveness of the MCD algorithm (Fauconnier & Haesbroeck, 2009). To facilitate our selection, the MCD is applied to *5* different values of α starting at the default value of *0.5*, and up to *0.9*, with increments of *0.1*. Table 5.2 shows the results of the different α-values in terms of the number of games that are tagged—as a result of both the corresponding α-value and the corresponding measurement—as outliers.

The results in Table 5.2 clearly show that *FS* as a measurement appears to be less affected by the potentially different/varying mechanisms that are responsible for the presence of outliers. This holds except for the case of Casual games with an α-value of *0.5*, as using the *FS* measure results in no games being tagged as outliers. This α-value of *0.5* is convincingly inefficient—as expected—with the number of outlying games more than tripling in comparison to higher α-values. However, even when using higher α-values, the fact remains that the number of outlying games repeatedly fails to increase monotonically with the decreasing α-values which suggests masking and/or swamping effects (Davies & Gather, 1993). This may signify concerns if we relied on each of the available measurements 'independently' while detecting the outliers. Therefore, we added a fourth column per experience-level (i.e., *F & P*) which includes the number of

games tagged as outliers depending on the logical conjunction of the measurements *F* and *P*. That is, a game is tagged as an outlier iff it was considered an outlier using both *F* and *P* independently. *FS* is excluded from the conjunction due to its continuous inconclusiveness. The conjunction *'F & P'* appears more reliable, and thus will be selected—along with the α-value of *0.7*—as the concluding parameters of the outlier detection module. All the games tagged as outliers in line with both parameters are excluded from *trainDS*.

Table 5.2. The number of games tagged as outliers *(from a total of 127 games per experience)*, among the different measurements, and the corresponding MCD α-value. The finally selected α-value is highlighted in grey.

MCD Alpha	**The Number of Games tagged as Outliers**											
	Casual Games				**Intermediate Games**				**Advanced Games**			
	F	**FS**	**P**	**F & P**	**F**	**FS**	**P**	**F & P**	**F**	**FS**	**P**	**F & P**
0.5	56	56	56	56	21	0	19	14	22	0	12	9
0.6	15	0	19	12	17	0	14	10	17	0	17	12
0.7	12	0	22	10	17	0	11	9	13	0	11	8
0.8	14	0	16	10	14	0	9	7	11	0	10	8
0.9	8	0	11	5	8	0	7	6	9	0	10	8

A final conclusion that can be drawn from Table 5.2 is the obvious advantage of *FS*—compared to *F* and *P*—when it comes to outliers. This can be partially explained by the fact that *FS* is the only measurement that 'accurately' takes into consideration the actual size/length of a game (i.e., the number of moves). The *F* simply counts the occurrences of the reasons and totally ignores the length of a game. On the other hand, *P* implicitly takes a game's length into consideration by reporting the fractions of the total occurrences (namely percentages), and thus is proportional to the number of moves

per game. Still, the *P* measurement experiences an inconsistency due to the overlapping nature of the aggregated reasons subsets; as single reasons are frequently included in two subsets. For instance, since the reason *'owl-attacks X'* attacks the board intersection *X* while using the *owl* module (See Section 5.2), this reason is included in both the *'Considered an Attack'* and the *'Thoughtful'* subsets. To further investigate the possibility of the game's length as a justification, Table 5.3 shows the statistics of the games' length between the games tagged as outliers/not-outliers. The *Median* and the *Median Absolute Deviation 'MAD'* (Davies & Gather, 1993) are chosen as robust univariate measures in case the dataset is still contaminated by outliers. The columns *'> Median'* and *'< Median'* are the medians calculated for outlier games with a length respectively greater/smaller than the not-outliers median. Table 5.3 shows a significant difference in the medians for the lengths of the games tagged as outliers in comparison to those tagged as not-outliers, particularly for the *Casual* games.

Table 5.3. The games' lengths for the outliers/not-outliers, and across the varying experiences.

Casual Games				**Intermediate Games**				**Advanced Games**			
Not Outliers		***Outliers***		***Not Outliers***		***Outliers***		***Not Outliers***		***Outliers***	
Median	**MAD**	**> Median**	**< Median**	**Median**	**MAD**	**> Median**	**< Median**	**Median**	**MAD**	**> Median**	**< Median**
245	22	279	228	273	17	306	266	269	21	288	257.50

5.3.3 Statistical Analysis

After defining the measures to be calculated and the sets used to group those measurements, the final step is to analyse the grouped measurements using simple statistical measures *(the Median and the Median Absolute Deviation 'MAD')*, followed by statistical hypothesis testing. The statistical measures, previously defined by Equations 4.3 and 4.4, will now be redefined according to the new features extracted. Given the set of experiences $E = \{e_1, e_2, \ldots e_n\}$, let D_e denote a subset of the dataset of

all games D where the experience of both opponents is e. The median can be estimated as:

$$Median_{e,s,\varphi} = \left(\varphi_{\lfloor(|D_e|+1)/2\rfloor:|D_e|,R_s} + \varphi_{\lfloor|D_e|/2\rfloor+1:|D_e|,R_s}\right)/2 \qquad \text{(Eq. 5.4)}$$

where φ is the measurement function *(i.e. denoting F, FS, and P)*, R_s is the s*th* reasons subset, $|D_e|$ is the number of games in D_e, and $\varphi_{1:|D_e|} \ldots \varphi_{|D_e|:|D_e|}$ are the order statistics of $\varphi_1 \ldots \varphi_{|D_e|}$. Accordingly, MAD can be estimated as:

$$MAD_{e,s,\varphi} = Median\begin{pmatrix} \left|\varphi_{1:|D_e|,R_s} - Median_{e,s,\varphi}\right|, \ldots \\ \ldots, \left|\varphi_{|D_e|:|D_e|,R_s} - Median_{e,s,\varphi}\right| \end{pmatrix} \qquad \text{(Eq. 5.5)}$$

For a second time, our main argument is that the medians of the different reasons subsets can model how the general strategy is decomposed into characterizing sub-strategies, in addition to demonstrating the variations in the strategies employed by the human Go players of different experiences. To confirm the potential hypotheses suggested by the data, both a two-sample T-test and a two-sided Wilcoxon rank sum test were once more used. By permuting the reasons subsets, the calculated measurements, and pairs of different experiences, the T-test and Wilcoxon-test will examine the null hypothesis that the data *(i.e. measurements per game)* are with equal means/medians 'respectively' against the alternative that the means/medians are not equal.

5.3.4 Discussion

In the first phase of the experiments, we have run our system using the selected case-base of games that were played by *casual*, *intermediate*, and *advanced* human players. GNU Go as our reference point, replayed the games and generated reasons for the majority of the moves, thus estimating the strategical reasoning originally employed by the human players. The Frequencies (Eq. 5.1), Frequencies per Step (Eq. 5.2), and the Percentages (Eq. 5.3), were applied as measurements for aggregated subsets of the generated reasons per game. Subsequently, and between each distinct pair of

experiences, the Wilcoxon-test and a two-sample T-test were applied to statistically signify the ability of the calculated medians/means to differentiate between the corresponding distinct pair of experiences. Table 5.4 shows the medians (Eq. 5.4) and median absolute deviations (Eq. 5.5) among the *127* games per experience level.

Table 5.4. The Medians and Median Absolute Deviations (MAD) of the Different Subsets, Among Diverse Experiences, and Using Three Different Measurements.

Measurements	Reasons' Subsets	Casual Players		Intermediate Players		Advanced Players	
		Median	MAD	Median	MAD	Median	MAD
Frequencies *(F)*	*Not Recommended*	28	8	35	9	34	7
	Considered An	262	80	337	112	357	84
	Considered A	400	79	471	84	484	72
	Explicit Gains	123	13	138	13	139	14
	Thoughtful	134	40	175	47	189	44
	End of the Game	0	0	1	1	2	1
	All Reasons	840	174	1021	191	1053	163
Percentages *(P)*	*Not Recommended*	3.49345	0.54313	3.21782	0.41675	3.22291	0.37645
	Considered An	30.8965	3.54481	33.2651	3.89565	33.6314	2.68176
	Considered A	46.6437	1.64371	45.6499	1.86296	45.7627	1.50347
	Explicit Gains	15.2370	3.43873	13.8358	3.52675	13.0590	2.96051
	Thoughtful	15.5794	1.64839	16.9267	1.55336	17.6814	1.46203
	End of the Game	0	0	0.11547	0.11547	0.15527	0.09492
	All Reasons	100	0	100	0	100	0
Frequencies Per Step *(FS)*	*Not Recommended*	0.12062	0.02429	0.12758	0.02576	0.12692	0.02250
	Considered An	1.03902	0.27431	1.26164	0.33599	1.32222	0.24774
	Considered A	1.62809	0.19564	1.73037	0.22134	1.80139	0.17270
	Explicit Gains [α]	0.50889	0.06054	0.51528	0.06837	0.51546	0.06208
	Thoughtful	0.55272	0.12294	0.65185	0.14344	0.70034	0.12554
	End of the Game	0	0	0.00387	0.00387	0.00666	0.00361
	All Reasons	3.41811	0.46188	3.81016	0.57400	3.94117	0.46279

All the values reported in this table are calculated using *127* games per each level of experience.

[α] The only rows where the subset and the applied measurement could not differentiate in a statistically significant manner was between the casual/intermediate and casual/advanced players. All other rows in this table show a case where a statistical significant difference in the medians and means between casual/intermediate and casual/advanced players was found (using the Wilcoxon-test and T-test respectively). None of the available medians/means show a statistically significant difference between intermediate and advanced players.

As opposed to the connectivity-patterns presented in Chapter 4, almost all of the sets of reasoning succeeded in significantly differentiating between *2* pairs of expertise-levels. Table 5.4 shows a statistical difference between casual/intermediate and casual/advanced human Go players, yet it fails to differentiate statistically between

intermediate/advanced players. The medians tend to get higher with experience considering both *F* and *FS* as measurements; the only exceptions are when the medians of the advanced are lower than or almost equal to the corresponding intermediate in both subsets *Not Recommended* and *Explicit Gains*. Though this apparent correlation between the *F*/*FS* medians and the growing experience is expected to some extent because more experienced players tend to play longer games, the two previously mentioned cases highlights the possibility that more experienced human players are less attracted by direct/instant gains and are more considerate when it comes to not recommended moves. This possibility is supported by medians reported for the measurement *P*, where the medians of both subsets almost consistently decrease with growing experience.

Using *P* again, the medians of the subset *Considered A Defence* somewhat decreases with growing experience, suggesting that a more aggressive strategy is applied by well-experienced human players as opposed to a more defensive strategy by their less-experienced counterparts. The later suggestion is supported by the medians reported for the subset *Considered An Attack* which increase in correlation with growing experience in view of all of the three measurements. The medians reported for both subsets *Thoughtful* and *End of the Game* also increase in correlation with growing experience for all of the three measurements.

Thus we can generally claim that, with rising experience, a human player's strategy evolves to a more thoughtful and aggressive strategy, a strategy that cares more about the final steps and eludes the not recommended moves, and last but not least, a strategy that is less lured by direct gains. This claim is statistically supported for human players who progress from casual to both intermediate and advanced experiences.

Though the inability to differentiate between experiences can be expected due to any inherent limits in an AC_R, this inability can also be caused by the fact that strategies cannot be merely estimated by the reasons behind them; there are other facts such as the timing of when a good-reasoned move is played. However, the absolute deviations

presented in Table 5.4 can still provide us with an insight on the difference between intermediate and advanced players. In general, the absolute deviations between intermediate players are higher than both casual and advanced players. As for the advanced players, their deviations are in general slightly higher than casual players for *F* and *FS* measurements, yet they are constantly lower than casual players using *P*. This finding may suggest that progressing from casual to intermediate experiences leads to more varying personal style(s) which become more conformed while progressing to a more by the advanced level of experience.

Finally, from the analysis of the reported results, the explanatory in-depth situation reasoning that was provided by the selected assessment component AC_R is capable of linking a particular board position to the consequent executed action. Consequently, the proposed methodology successfully characterized the differences in the expertise levels – of the human Go players – in an automated and passive approach. In addition, the description of those differences was quantitatively-represented and statistically-significant.

5.4 A Case Study: Tracking the Reasoning Competencies

In the second and final phase of our experiments, we created our proposed classifier. Then, using the *testDS*, the games of each player were temporally ordered and classified. The resulting classification probabilities are then cumulatively averaged to generate *Monitoring-Curves* to observe the learning activity per player. This leads to the final stage, in which we diagnose this learning activity by temporally observing each of the strategies' characteristics *(i.e., the aggregated subsets of reasons)*. The *testDS* for this phase is the one that was proposed in Section 4.4.2. To recap: from the *No Name Go Serve* (NNGS), *400* games were initially selected, played by *246* distinct registered-names. A threshold of at least *10* games was imposed to select the final cases, and this yielded a final set of *15* players (Table 4.5).

The classifier architecture for this phase is also the one previously proposed (see Figure 4.5); a three-tier ensemble with the capability to predict the class label of a game

of Go as *Casual*, *Intermediate*, or *Advanced*. The first-tier is based on Random Forests (RFs), which are ensembles of Classification Decision Trees (CDTs). Each individual classifier (i.e., RF) is trained to classify a class and its complement, outputs two probabilities, and is created using a maximum number of decision trees (set to *1000* throughout this study), yet only the number of trees corresponding to the minimum cumulative misclassification probability (Figure 4.6) is used while testing. The third and final tier combines the results from the two previous tiers, and using a final gate-function it creates – for each game – a single probability per class (Eq. 4.7).

5.4.1 Experimental Setup

As in the previously conducted experiments, RFs were used which had a maximum of *1000* classification decision trees, and a minimum number of observations per tree leaf of *1*, and a forest's attributes were determined according to the *Error* and *Size* measures. In addition to the ensemble errors, True Positive (TP) and True Negative values — estimated using a five-fold cross-validation — were also calculated. Using the uncontaminated *trainDS*, *30* random forests were trained to differentiate between each experience level and its complement, employing the *3* measurements. Table 5.5 shows the out-of-bag misclassification probabilities, true positive and true negative values.

The results are adequate to estimate the experience of a human Go player, especially — as in the next step — when combined using a gate-function and applied to a history of games. The results are comparable to our previous findings (Table 5.4), and still show a better capability for classification between *Casual* and *Not-Casual* games, then by *Advanced* and *Not-Advanced*. Classifying *Intermediate* versus *Not-Intermediate* games reported the lowest accuracies; the highest ensemble errors and the lowest true positive/negative values.

Once more, Table 5.5 shows the competence of the measurement *FS* which clearly reports improved results when compared to *P*. Although the results of *F* are repeatedly better than those of *FS*, the differences there are not as significant as the case when *FS*'s results are better. Combined with its noticeable intrinsic ability to not be

affected by the mechanism(s) generating outlying observations, *FS* has been selected as the most reliable measurement to monitor the players' learning behaviour.

Table 5.5. The Ensemble Errors, Sizes, True Positives, and True Negatives of the random forests trained to classify between the experiences of human Go players. The lowest Ensemble Error and the highest True Positive and True Negative values for each experience and its complement are in bold. All the values reported are arithmetic means (averages) followed by standard deviations.

		Minimum Ensemble Error	Corresponding Ensemble Size	True Positive	True Negative
Casual Vs. Not Casual	F	**0.2382±*0.0197***	216.2667±*239.8535*	0.4852±*0.1026*	**0.8562±*0.0495***
	FS	0.2476±*0.0194*	250.8000±*262.8744*	**0.5370±*0.0947***	0.8494±*0.0423*
	P	0.2685±*0.0140*	201.6333±*232.6311*	0.4778±*0.0844*	0.8123±*0.0543*
Intermediate Vs. Not Intermediate	F	**0.3596±*0.0153***	154.9000±*261.6518*	0.2593±*0.1301*	**0.7426±*0.0997***
	FS	0.3733±*0.0194*	183.4000±*317.9671*	0.2469±*0.1587*	0.7049±*0.1308*
	P	0.3621±*0.0150*	141.1667±*212.0786*	**0.2679±*0.1432***	0.7426±*0.1140*
Advanced Vs. Not Advanced	F	0.3328±*0.0180*	298.2333±*313.6831*	0.3494±*0.0863*	0.7679±*0.0709*
	FS	**0.3215±*0.0178***	188.7667±*242.4006*	**0.3951±*0.0979***	**0.7889±*0.0684***
	P	0.3295±*0.0148*	185.0000±*212.8007*	0.3272±*0.0779*	0.7852±*0.0799*

5.4.2 Discussion

Using *testDS*, the games for each player – among the *15* finally selected – were temporally ordered and then replayed by the GNU Go engine to estimate the strategic reasoning behind the moves. *FS* was then applied – *and combined according to the aggregated reasons' subsets* – thus creating the final feature set for each game. The proposed classifier generate three final probabilities for each game, namely $Pr_{final}(C)$, $Pr_{final}(I)$, and $Pr_{final}(A)$. Per player, three *Monitoring-Curves* have been plotted; each

representing the 'un-weighted' Cumulative Moving Average (CMA) for an experience-level, with a maximum *window size* of *50* games (Eq. 4.8).

As in Chapter 4, for several pages a 2-D line graph is presented for each player. Each graph displays the value of the three *'Player's Experience Monitoring-Curves'*, in addition to displaying the *'Player's Rank'*, according to the online *No Name Go Server* archives. A label on the right *y*-axis represents the centre of the respective rank category. The *x*-axis displays the game number, with imposed temporal frames for the corresponding dates (months/years).

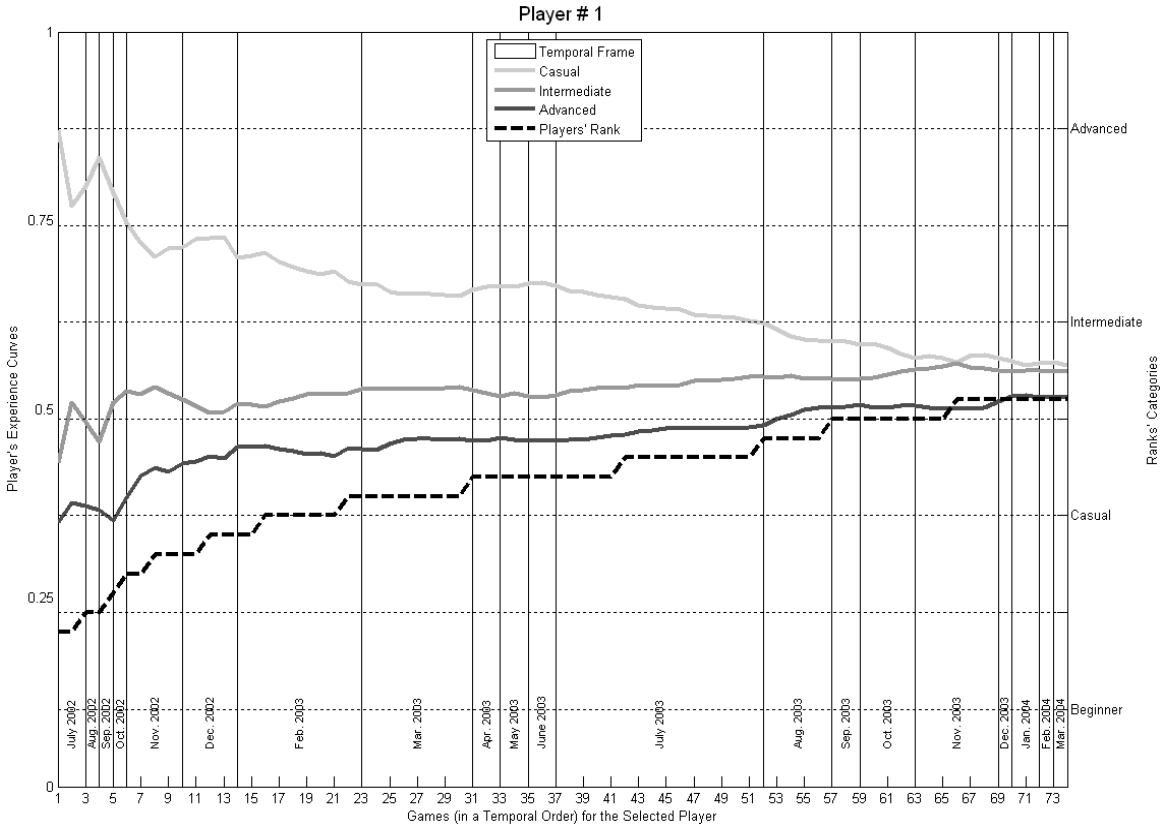


Figure 5.5 (a). The Competency-Level Monitoring Curve – Player #1; *Based on the situational reasoning of both opponents and the proposed three-tier classifier*.

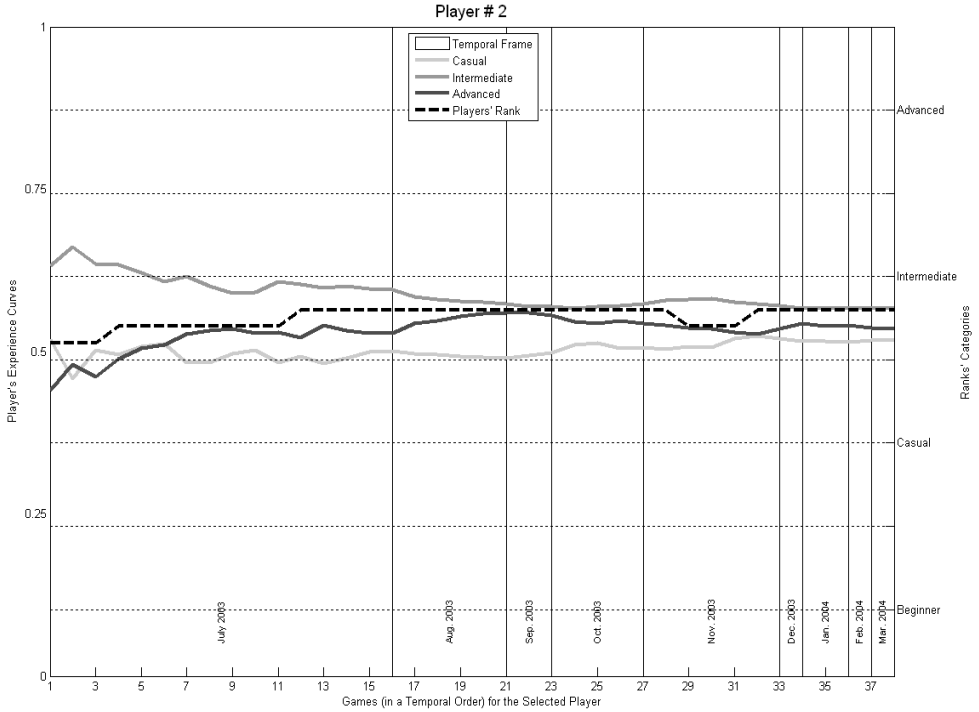


Figure 5.5 (b). The Competency-Level Monitoring Curve – Player #2; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

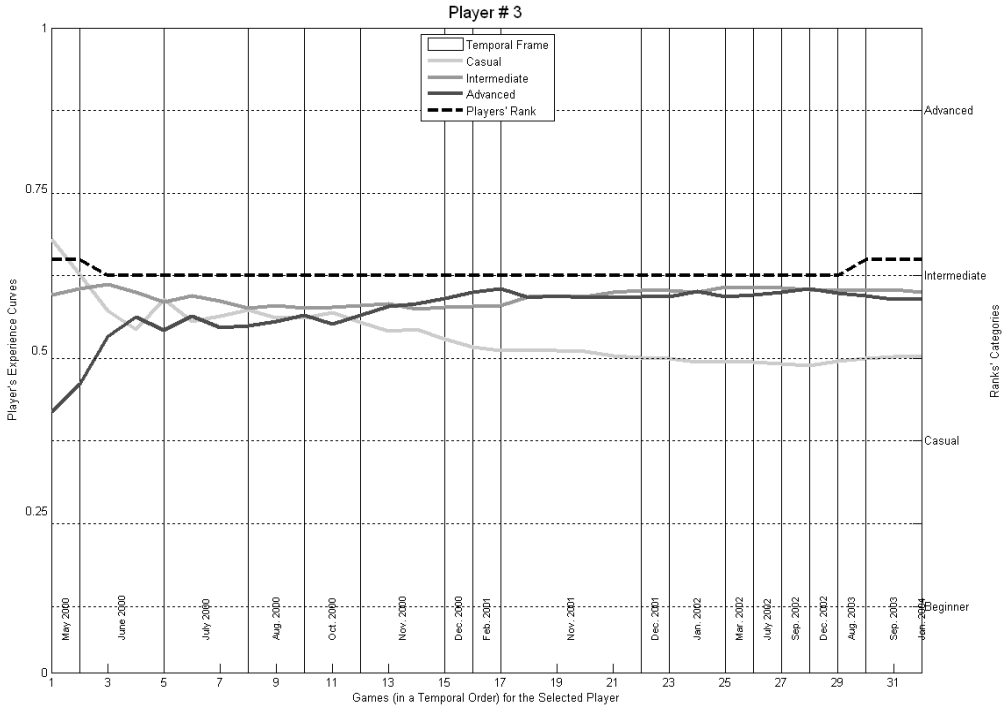


Figure 5.5 (c). The Competency-Level Monitoring Curve – Player #3; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

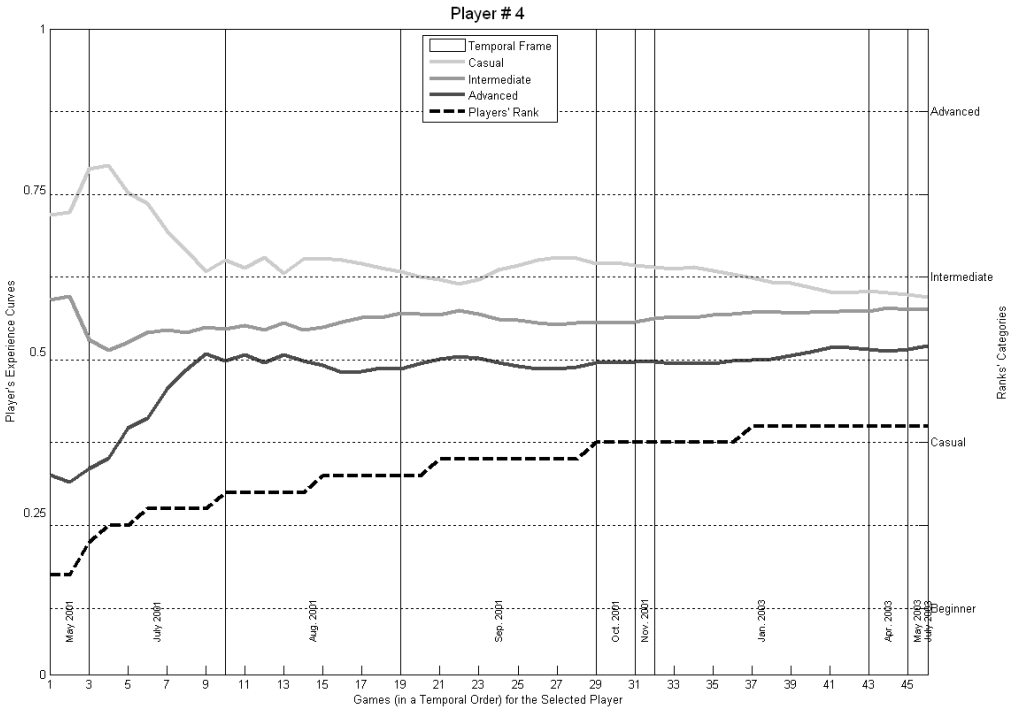


Figure 5.5 (d). The Competency-Level Monitoring Curve – Player #4; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

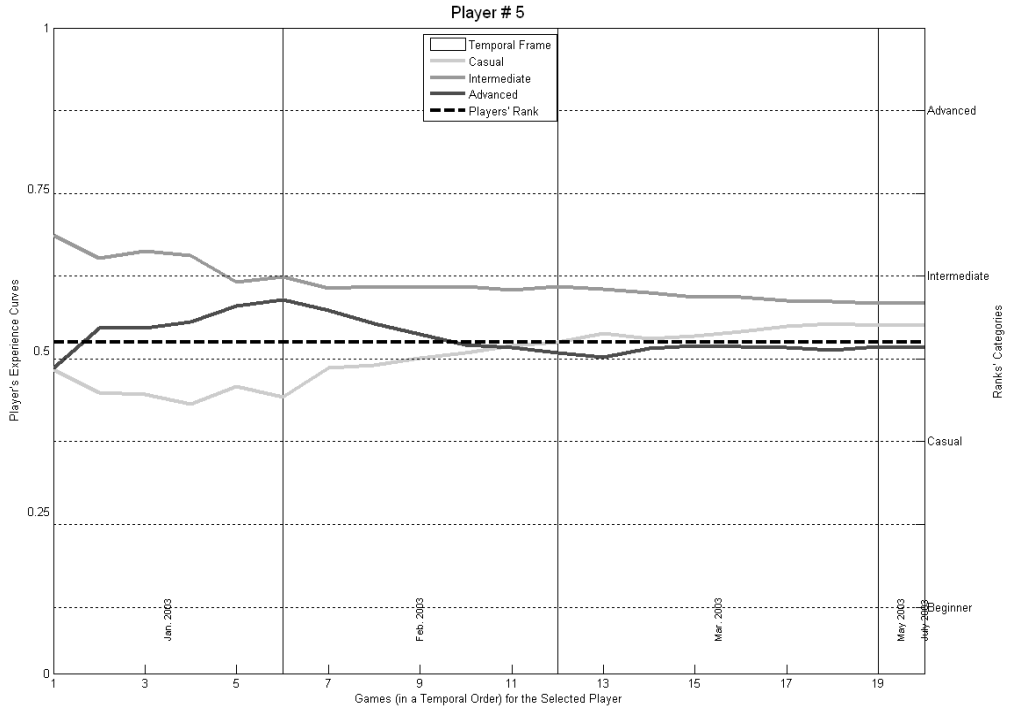


Figure 5.5 (e). The Competency-Level Monitoring Curve – Player #5; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

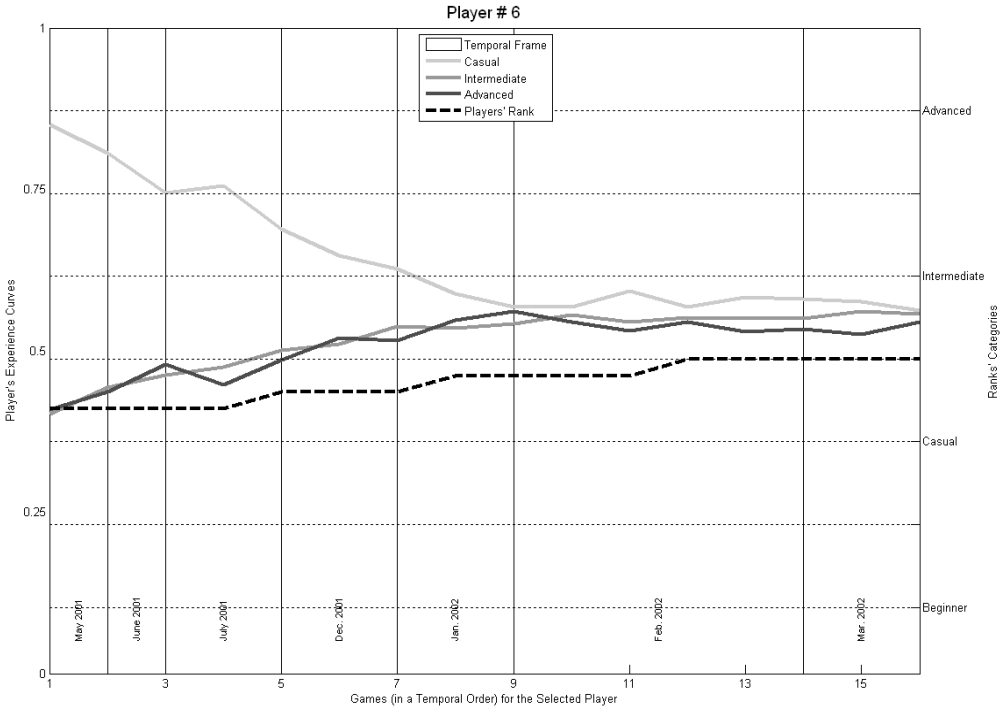


Figure 5.5 (f). The Competency-Level Monitoring Curve – Player #6; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

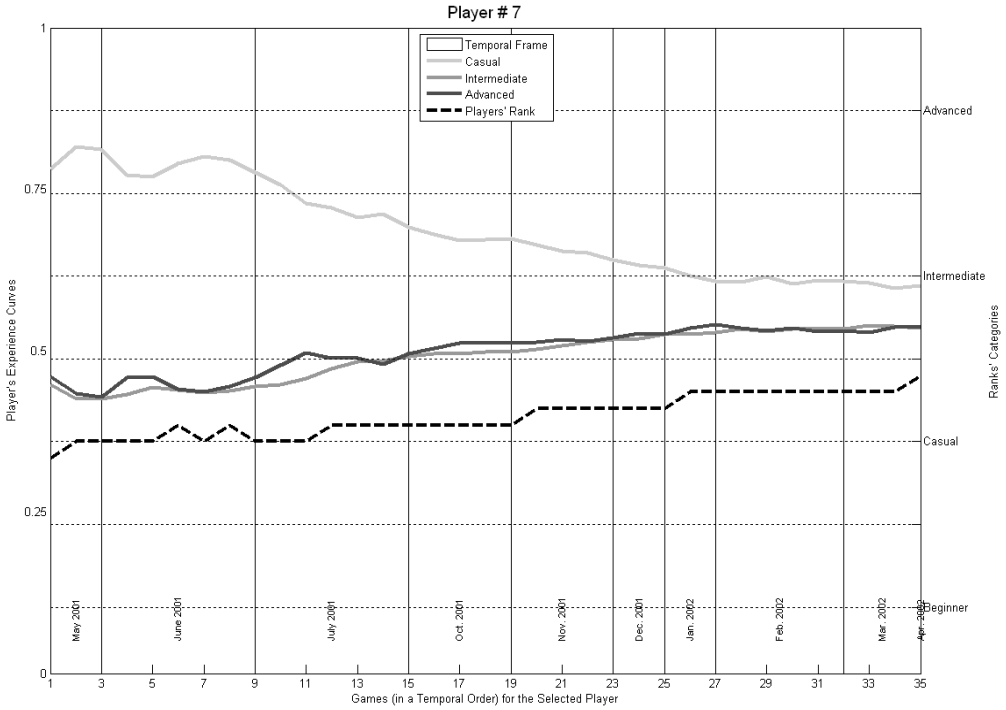


Figure 5.5 (g). The Competency-Level Monitoring Curve – Player #7; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

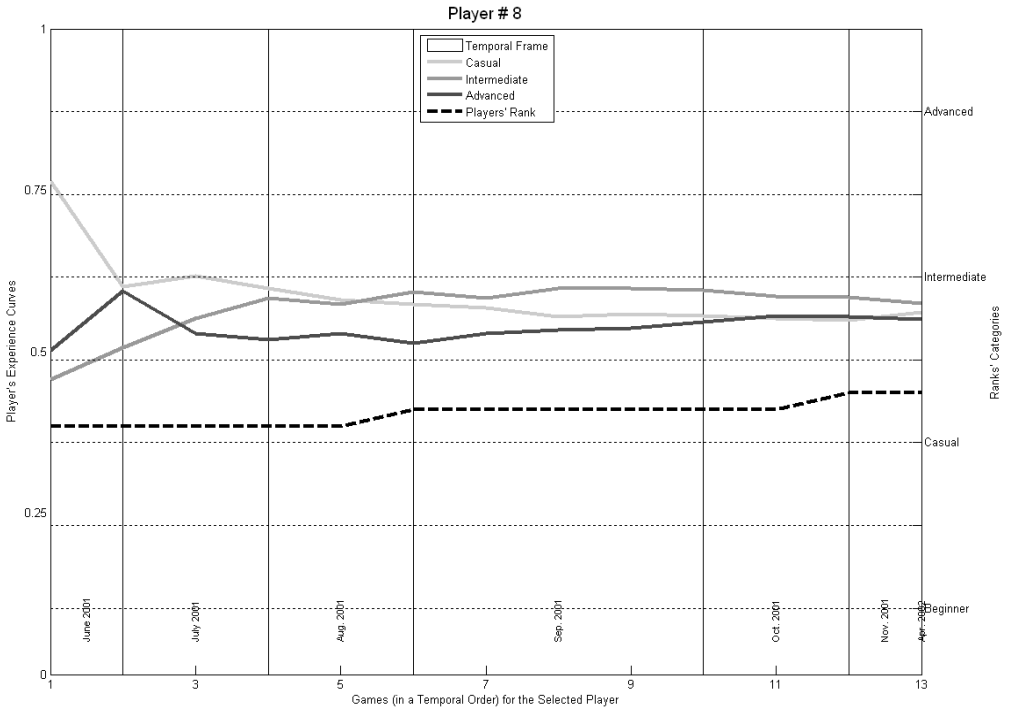


Figure 5.5 (h). The Competency-Level Monitoring Curve – Player #8; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

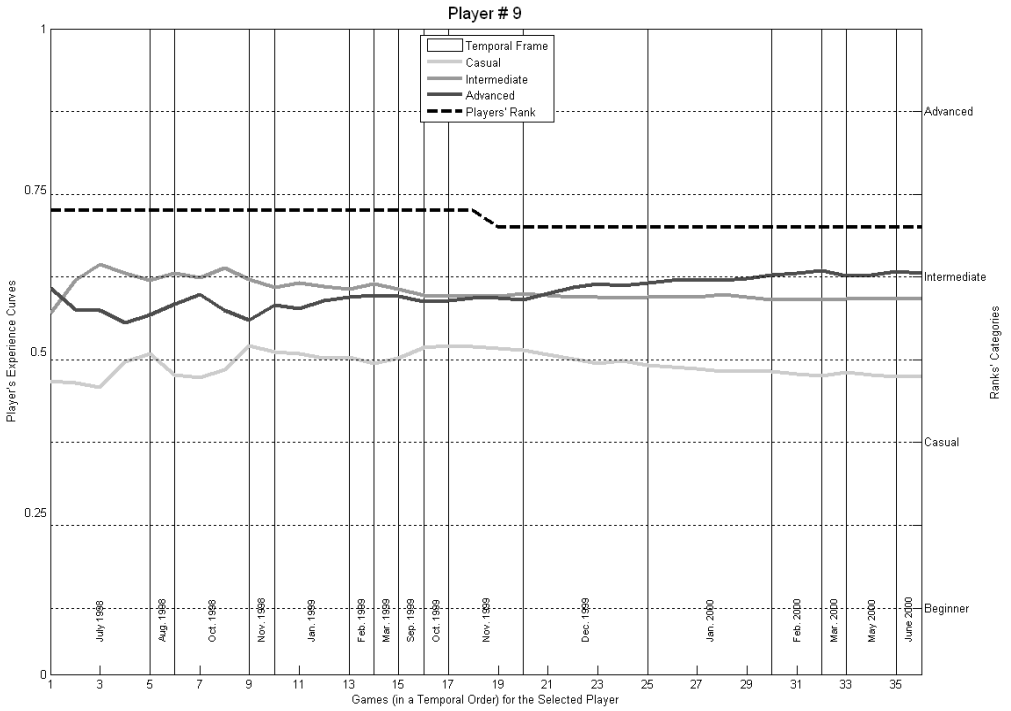


Figure 5.5 (i). The Competency-Level Monitoring Curve – Player #9; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

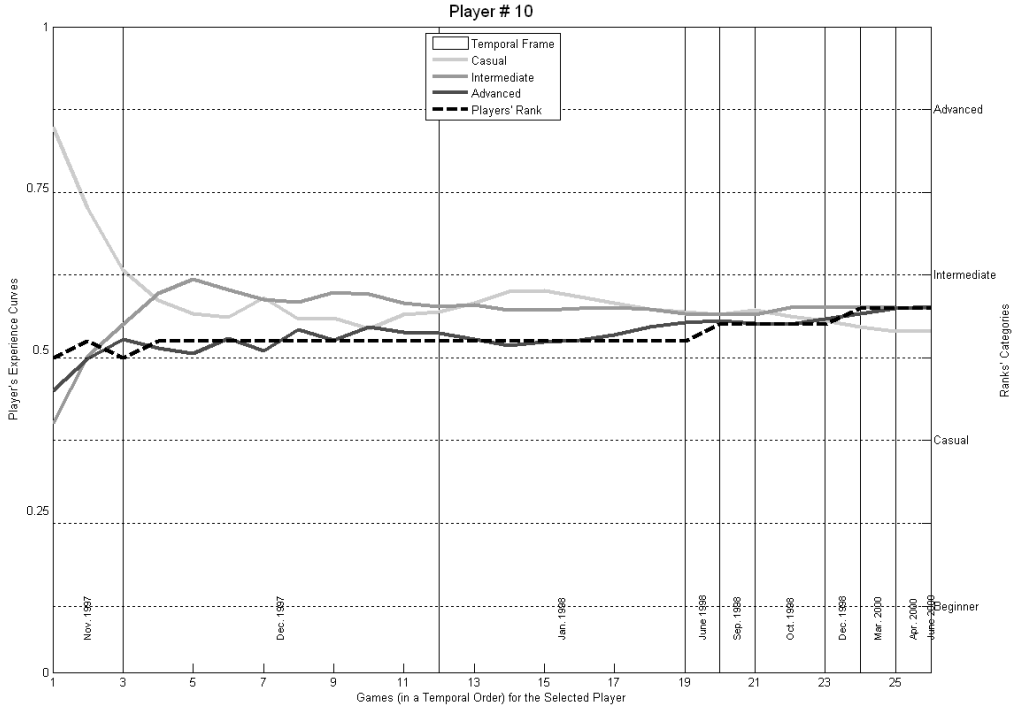


Figure 5.5 (j). The Competency-Level Monitoring Curve – Player #10; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

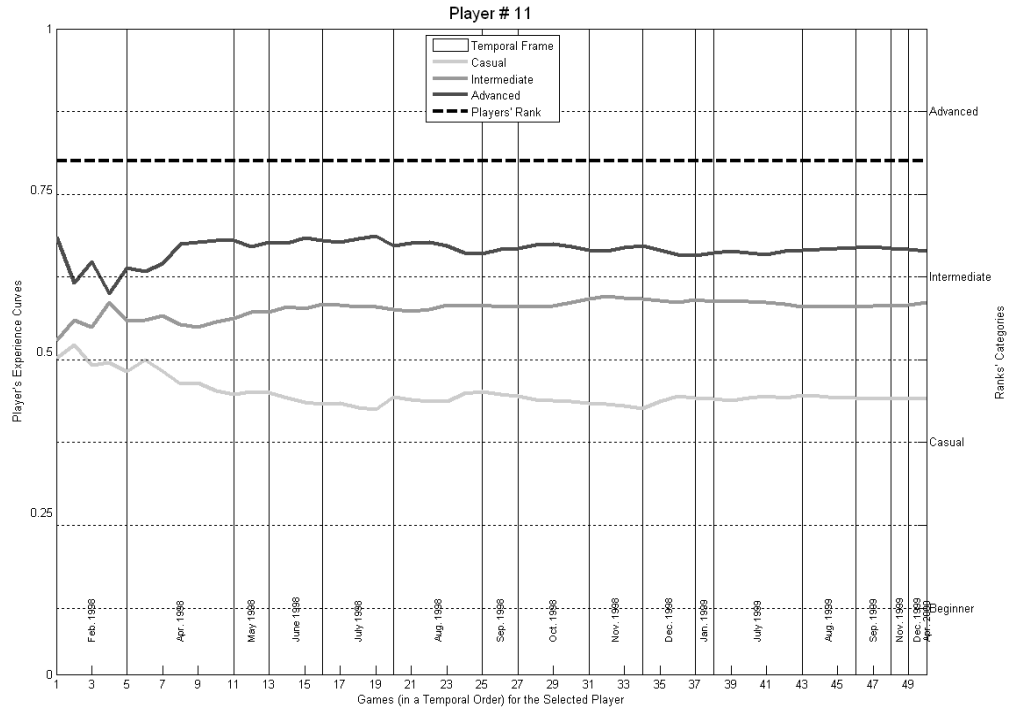


Figure 5.5 (k). The Competency-Level Monitoring Curve – Player #11; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

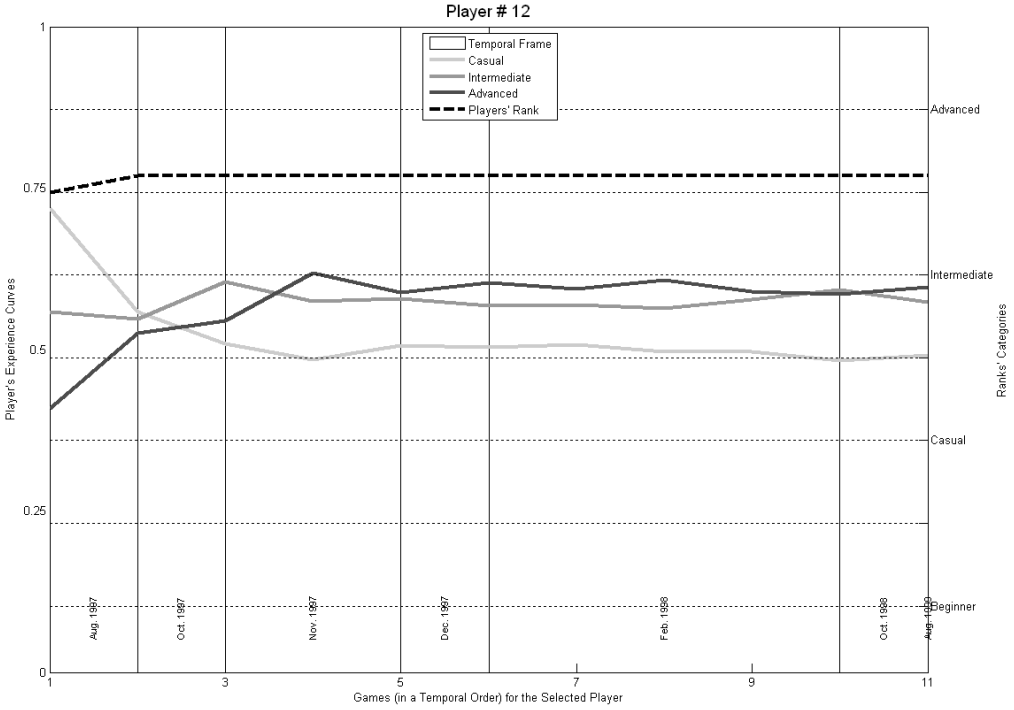


Figure 5.5 (l). The Competency-Level Monitoring Curve – Player #12; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

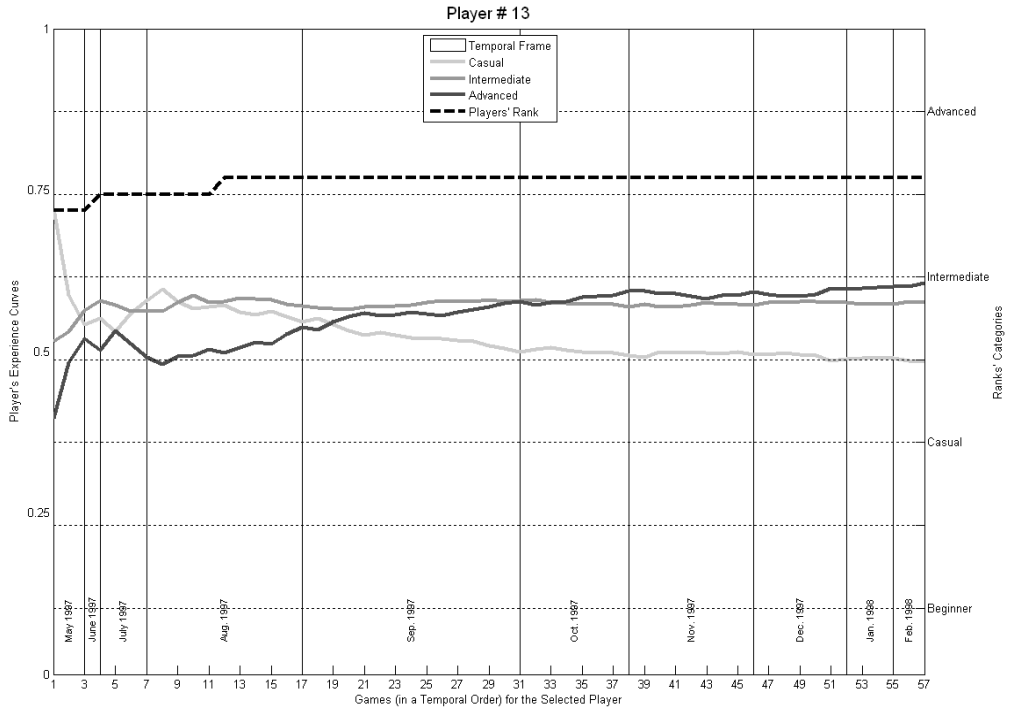


Figure 5.5 (m). The Competency-Level Monitoring Curve – Player #13; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

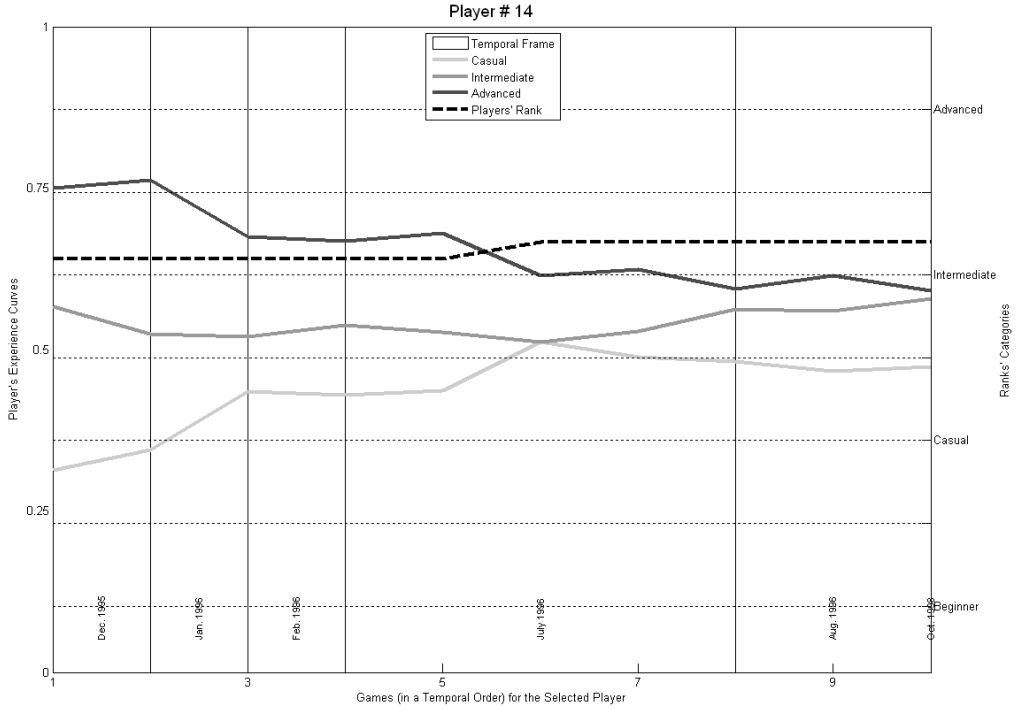


Figure 5.5 (n). The Competency-Level Monitoring Curve – Player #14; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

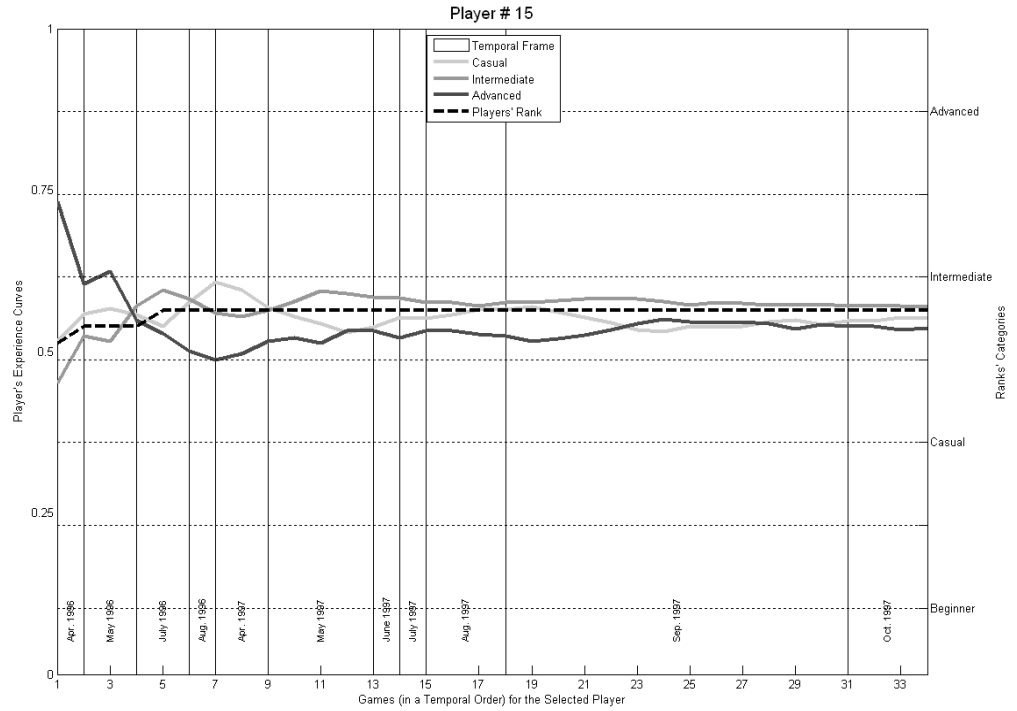


Figure 5.5 (o). The Competency-Level Monitoring Curve – Player #15; *Based on the situational reasoning of both opponents and the proposed three-tier classifier.*

The monitoring-curves in all of the resulting figures show a clear consistency between the experience level of a player and his/her probabilities' curves. That is, as the player 'assumingly' gains more experience with time, the probabilities' curves reflect this learning activity by either declining or rising. For instance, "Player #1" advances from an Upper-Beginner to a Lower-Intermediate experience over the course of the *74* games selected. Concurrently, the Causal-monitoring-curve of the mentioned player declines from *0.875* to around *0.5625*, the Intermediate-curve also converges to around *0.5625* rising from *0.4375*, and the Advanced curve is also rising from *0.3750* to a little bit higher than *0.5*. Although, on strict classification bases, this player is classified as a Causal during the whole period *(since the Casual probability curve is higher than both the Intermediate and Advanced curves)*, the clear trend in the curves assures that—with more games—the player is going to be correctly classified as Intermediate. Obviously, classifying a Beginner player as a Casual in this context is considered accurate, since—as we have mentioned in Subsection 4.3.1—no Beginner cases were included in the training dataset.

In this context, we would like to point out that even though the classifiers are trained using cases only from the mid-range of an experience level—in order to minimize the 'strategical' overlapping between the different levels—this is not the only reason for misclassifying games from around the boundaries between ranks. Alongside the potential personal-influences, we also refer to the supposition that expecting from an expert—for instance—a consistent performance at a particular level in all subtasks can be a mistake (Klein and Hoffman, 1993). Thus, a player who advanced from being a Casual to the Intermediate level is not expected to show this level of Intermediate-like proficiency in all aspects of the game. In addition, the specific expertise this player may have acquired might not be reflected by the features used in classification (i.e., the GNU Go estimated reasons). However, we are not ruling out other reasons for such misclassification; such as the limitations of the GNU Go engine, and the possible inconsistencies in the ranking methodology used by the online game-server.

Similar to "Player #1", the monitoring curve corresponding to the correct experience level is mostly higher than the other two curves for all of the players except players number *8*, *9* and *14*. The "Player #8" is a case where the player—in only *13* games—is advancing from Mid-Casual to Upper-Casual, and for the last *6* games, the probability of being an Intermediate is somewhat higher than that of being a Casual. An identical case is that of "Player #9", where the *36* games fall in the Upper-Intermediate experience level, and where starting from the 21^{st} game, the Advanced curve is slightly higher than the Intermediate. It is worth mentioning that both cases completely fall in the overlapping area between experience-levels, and that, in the case of "Player #8", the Casual curve is showing—at the very last games—an inclination of rising beyond the Intermediate curve. For "Player #14", the inclination of the Advanced curve to fall below the Intermediate curve can be observed from the beginning games. However, though the misclassification in cases number *8* and *14* can be attributed to the small number of games per player—*13* and *10* games, respectively—this is not shared by all the players with a short record. For instance, "Player #12", a Lower-Advanced player with a record of only *11* games, is in general correctly classified. Nevertheless, a considerable record of games notably leads to a more accurate monitoring, a fact demonstrated—in most of the figures—by the relatively sharp abrupt changes in the early-segments of the curves.

Those reported results demonstrate that the "Competency Model" developed using the quantitative measurements representing the explanatory in-depth situation reasoning is successfully capable of reflecting and monitoring the changes in the performance of a selected individual player. Additionally, as proposed, the monitoring approach is completely automated and passive, yet provides in-depth – *and declarative, due to the nature of the reasons* – domain-specific assessment.

Since a game of Go is an activity between two players—who even though our strict case-selecting conditions might have varying experience levels—we based the presented implementation on estimating the reasons behind the moves of both opponents in a game. However, an undeniable question would be: what if the

experiments are done using the estimated reasons behind the moves of *only* the selected player whom we are trying to monitor? That is, in all the games with a selected player in common, only the moves of this player are taken into account for creating the feature set. Certainly, this modification requires re-training the proposed classifier using only a single-player's features, i.e., each instance in the training dataset *trainDS* will be split into two, this resulting in *762* training instances rather than the original *381* instances.

Another compelling issue is about the computational effort wasted for creating the second-tier in the proposed classifier, i.e., the forest of RFs. Would it not be much simpler and computational effective if only the single best RF—the RF with the minimum out-of-bag classification error—is selected? Both questions are respectively answered in the following subsections.

5.4.3 Applicability to Analysing Players

The following figures show the results of using the estimated reasons behind the moves of sole players as features—for both training the classifier and monitoring the players' strategic-learning—rather than using the reasons behind the moves of both players (i.e., the whole games) as features as in the previously shown results. The monitoring-curves in all of the resulting figures show further uncertainty when compared to the previous results of considering the whole games in the creation of the features set. Commenting on the five players with the highest number of game-records (*previously selected in subsection 4.4.5*): "Player #1" still shows a decline in the Casual-monitoring-curve accompanied by a rise in both Intermediate and Advanced curves. However, the quantities of declination/rising exhibited are significantly lower than before. Using the games' features, this player showed an almost overlapping pattern between the Casual and Intermediate games starting from the 66^{th} game, while using the player's features the gap between both curves by the 74^{th} game remained considerable. Evidently, much more games are required for this player to be correctly classified as an Intermediate. Confusion can be also found in both players number *2* and *4*, with almost the entire temporal-frames of "Player #2' misclassified as an Advanced. "Player #4"

shows a misclassification between the games *11* to *14* — where the Advanced curve is higher than the Casual — that was not previously present.

In "Player #13", a visible distinction between the Casual and Intermediate curves can be seen in the beginning games, which can be thought of as an improvement when compared to the previous results. Conversely, for the latter games, the overlapping between the Intermediate and Advanced curves goes for a significantly longer period when compared to the previous results, an apparent disadvantage. "Player #11" is the only case where no substantial difference between using both features sets can be found, except for a noticeable overlap between the Intermediate and Advanced curves at game number *4*.

As in Chapter 4, and in order to provide a more quantitative interpretation of the competency-curves and their temporal trends with the changing rank, Table 5.6 will show the correlation between each competency-curve (Eq. 4.9) and the rank.

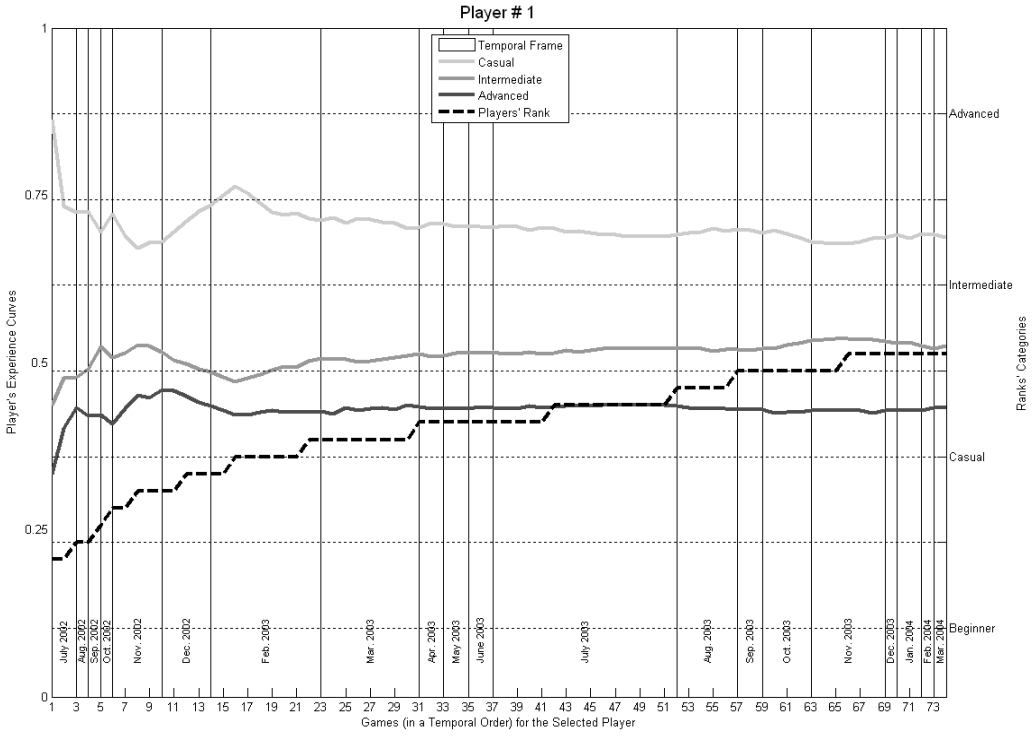


Figure 5.6 (a). The Competency-Level Monitoring Curve – Player #1; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

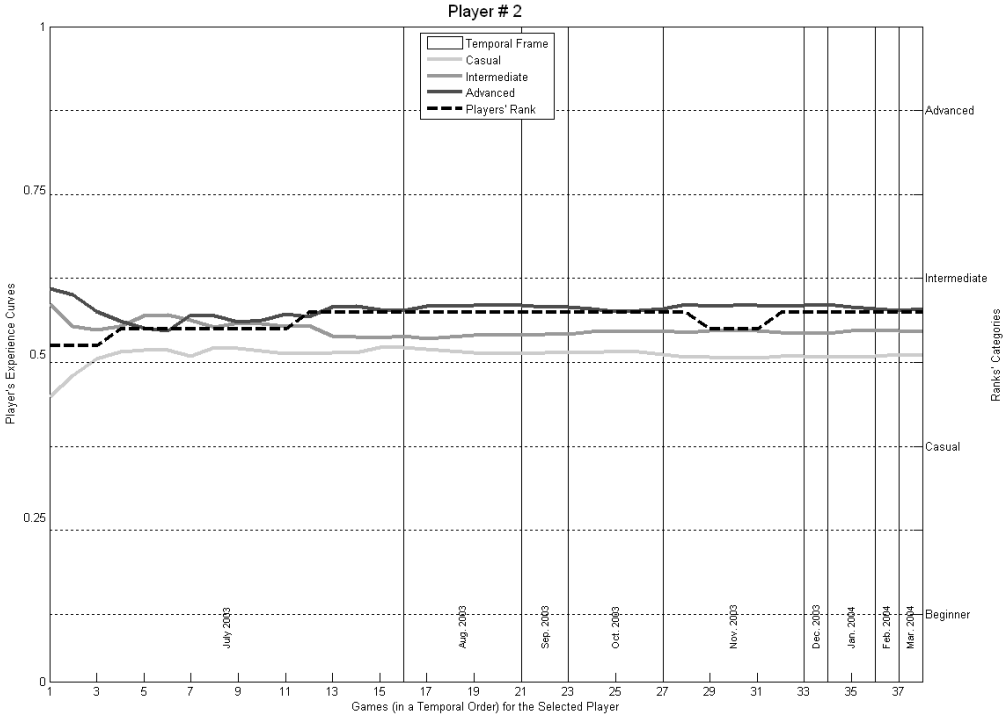


Figure 5.6 (b). The Competency-Level Monitoring Curve – Player #2; *Based on the situational reasoning of the individual player and the proposed three-tier classifier.*

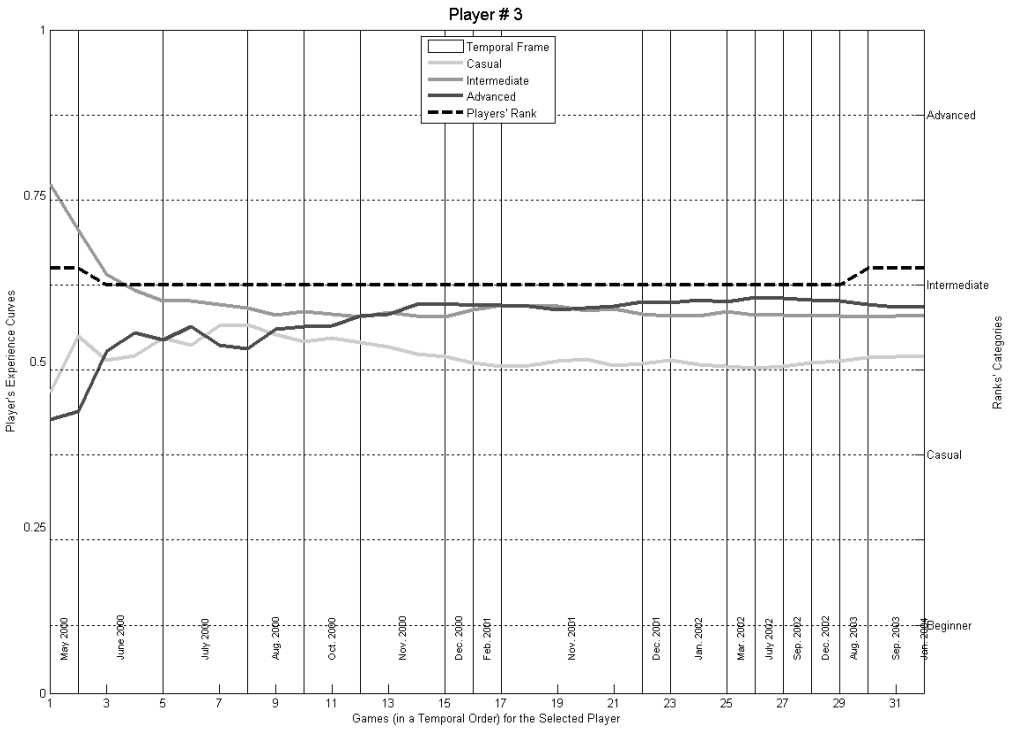


Figure 5.6 (c). The Competency-Level Monitoring Curve – Player #3; *Based on the situational reasoning of the individual player and the proposed three-tier classifier.*

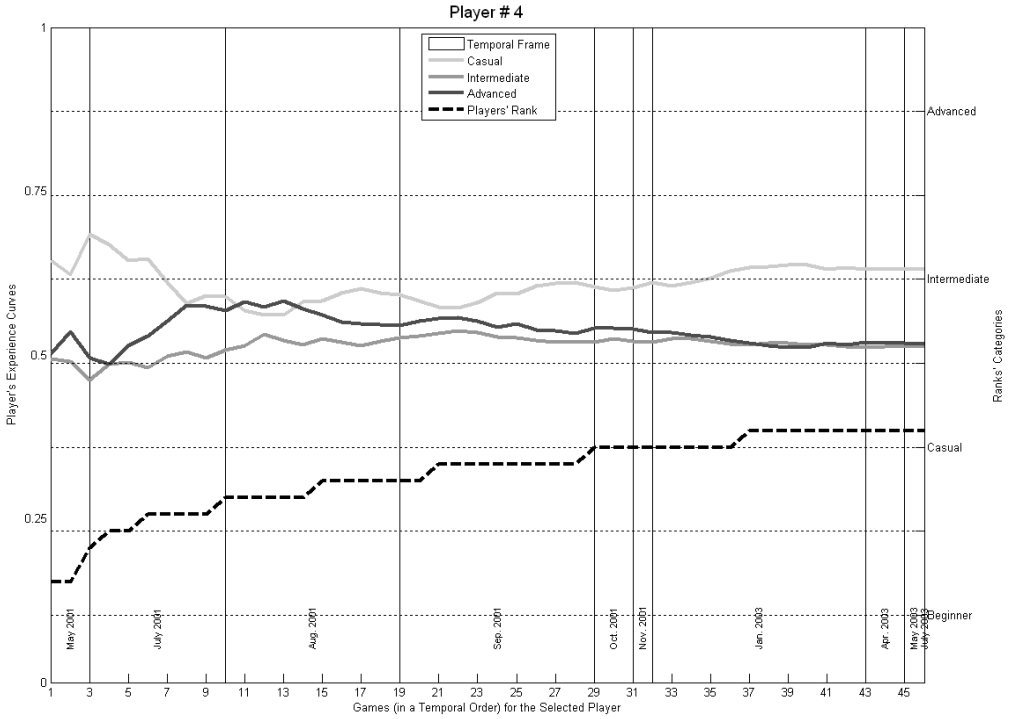


Figure 5.6 (d). The Competency-Level Monitoring Curve – Player #4; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

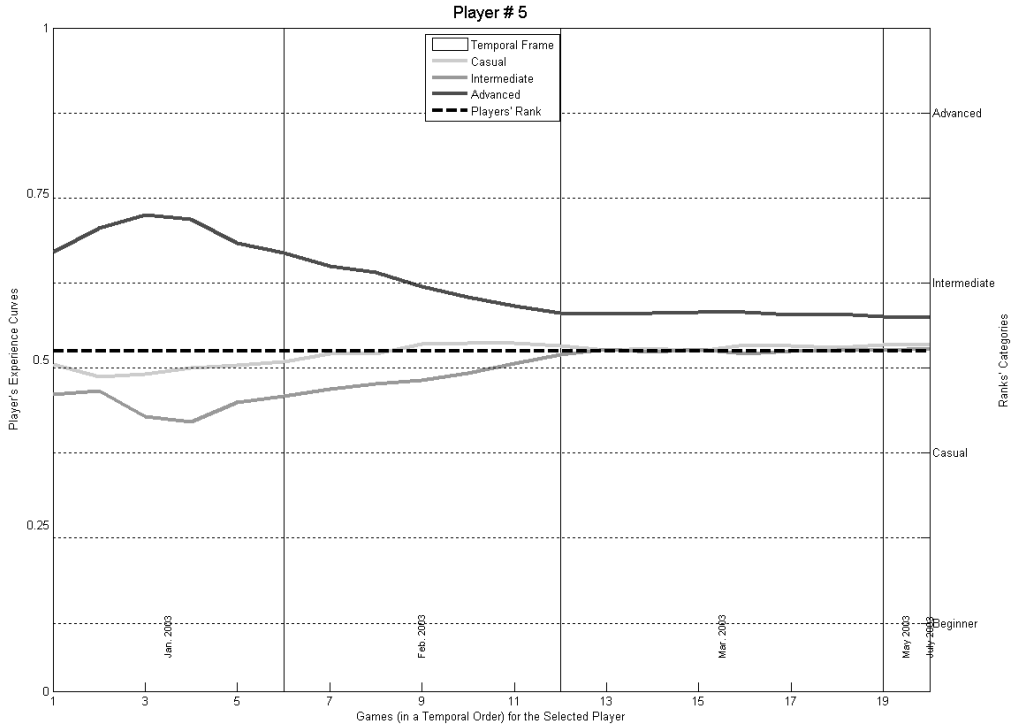


Figure 5.6 (e). The Competency-Level Monitoring Curve – Player #5; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

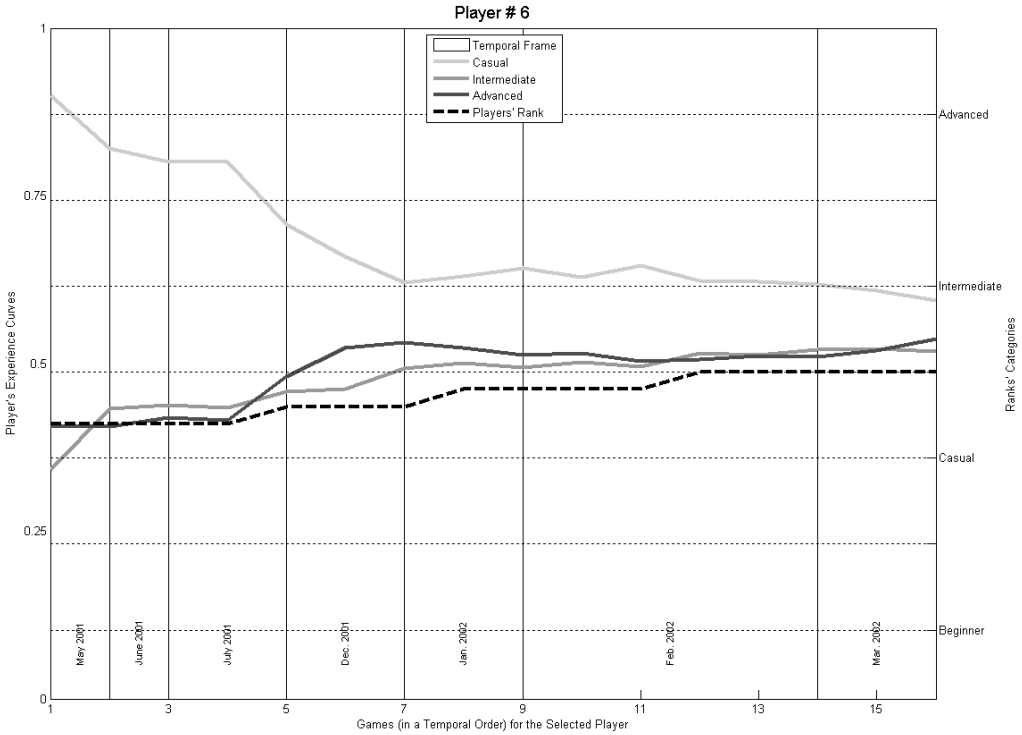


Figure 5.6 (f). The Competency-Level Monitoring Curve – Player #6; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

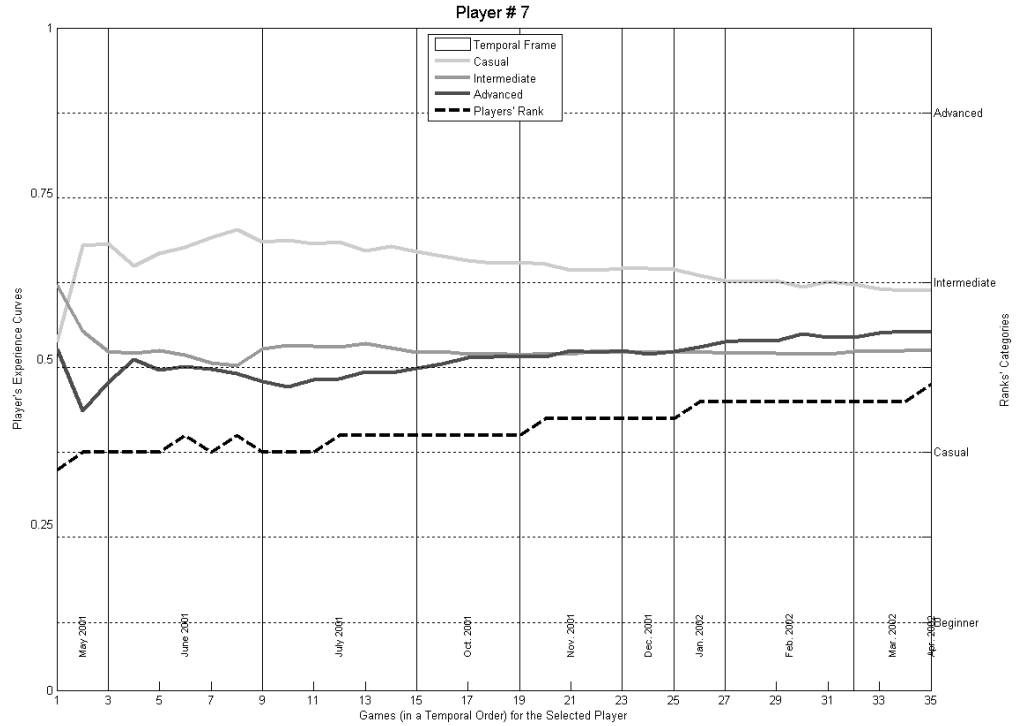


Figure 5.6 (g). The Competency-Level Monitoring Curve – Player #7; *Based on the situational reasoning of the individual player and the proposed three-tier classifier.*

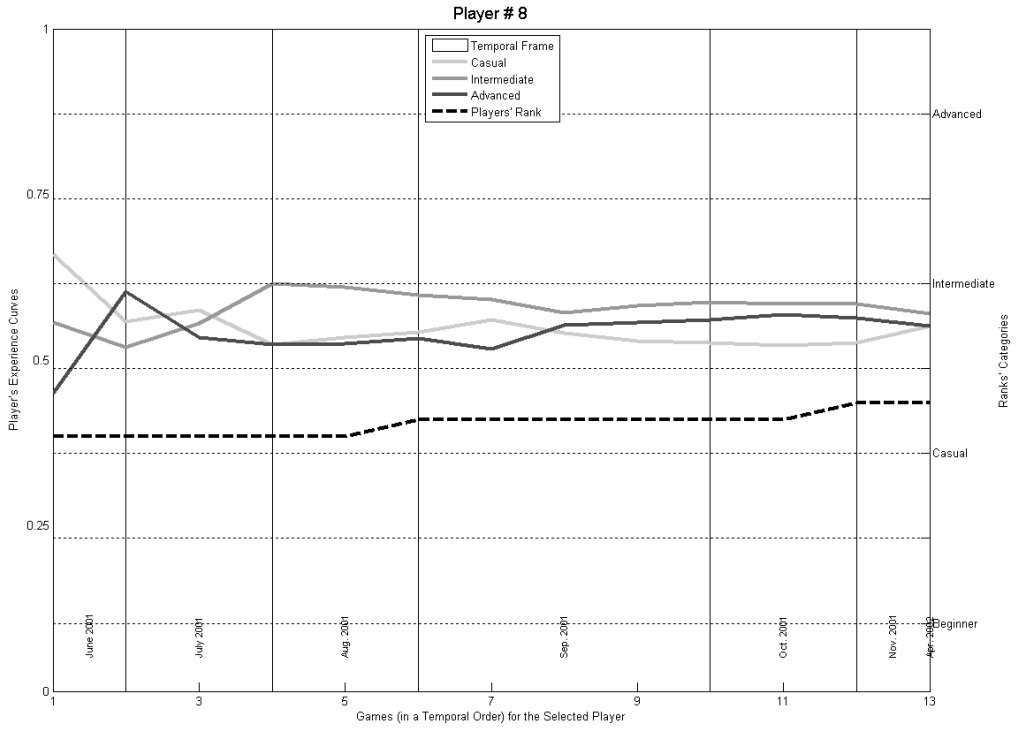


Figure 5.6 (h). The Competency-Level Monitoring Curve – Player #8; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

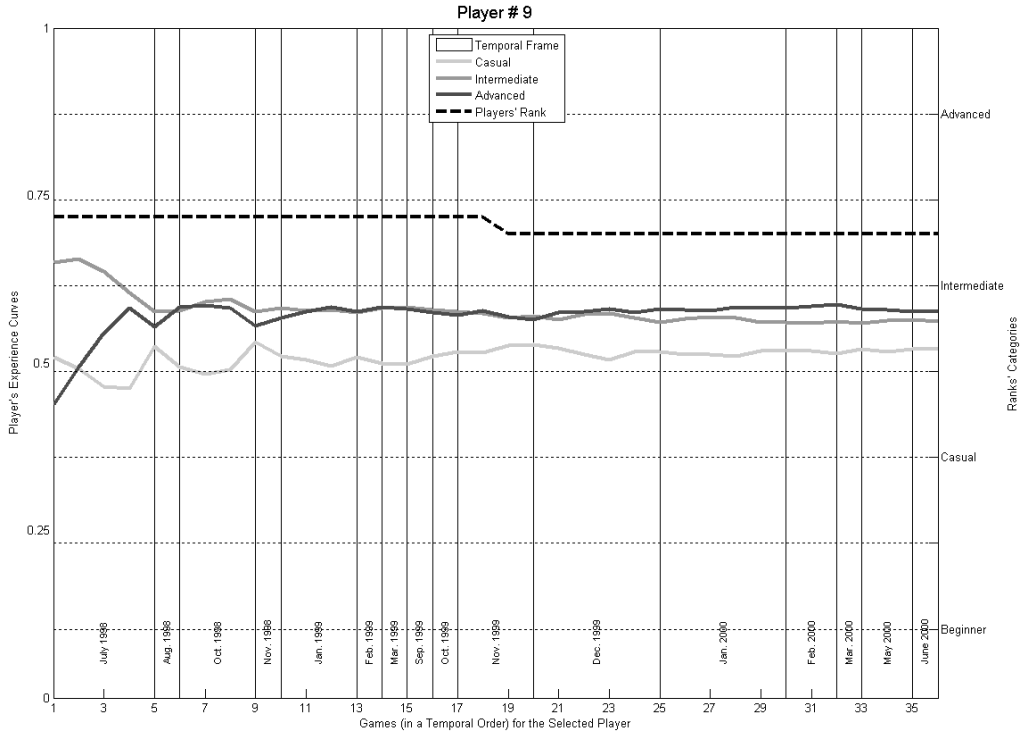


Figure 5.6 (i). The Competency-Level Monitoring Curve – Player #9; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

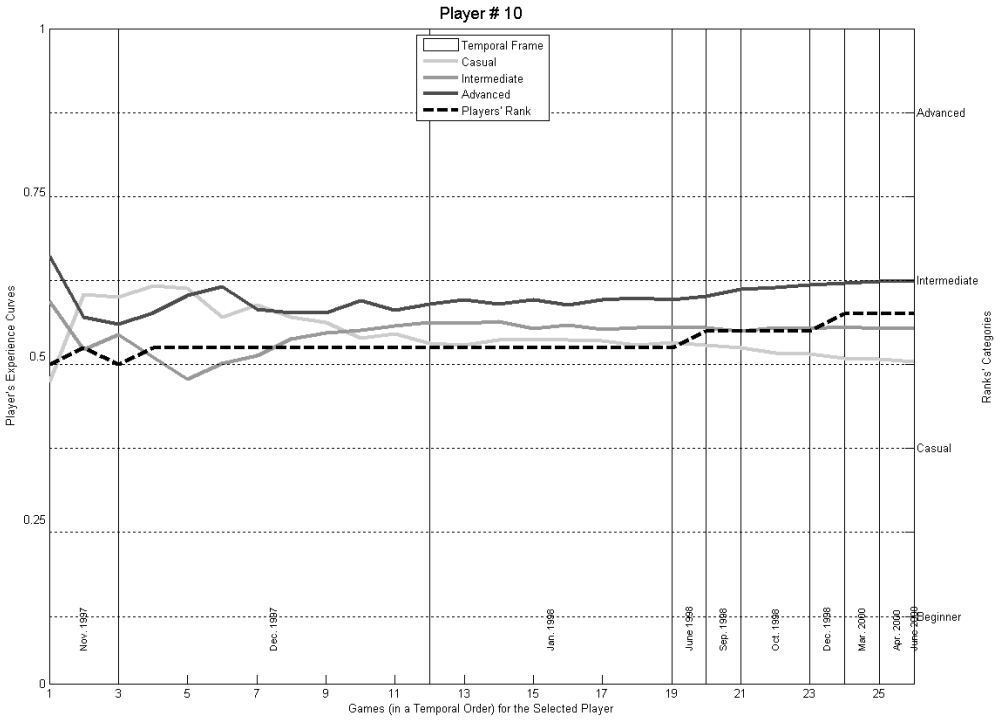


Figure 5.6 (j). The Competency-Level Monitoring Curve – Player #10; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

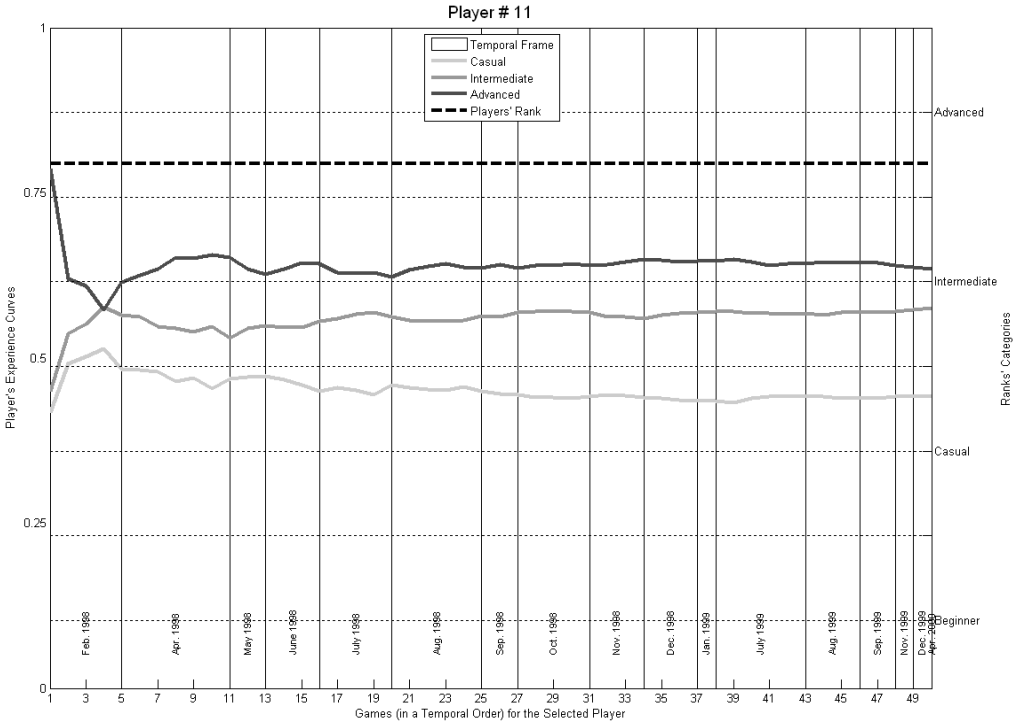


Figure 5.6 (k). The Competency-Level Monitoring Curve – Player #11; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

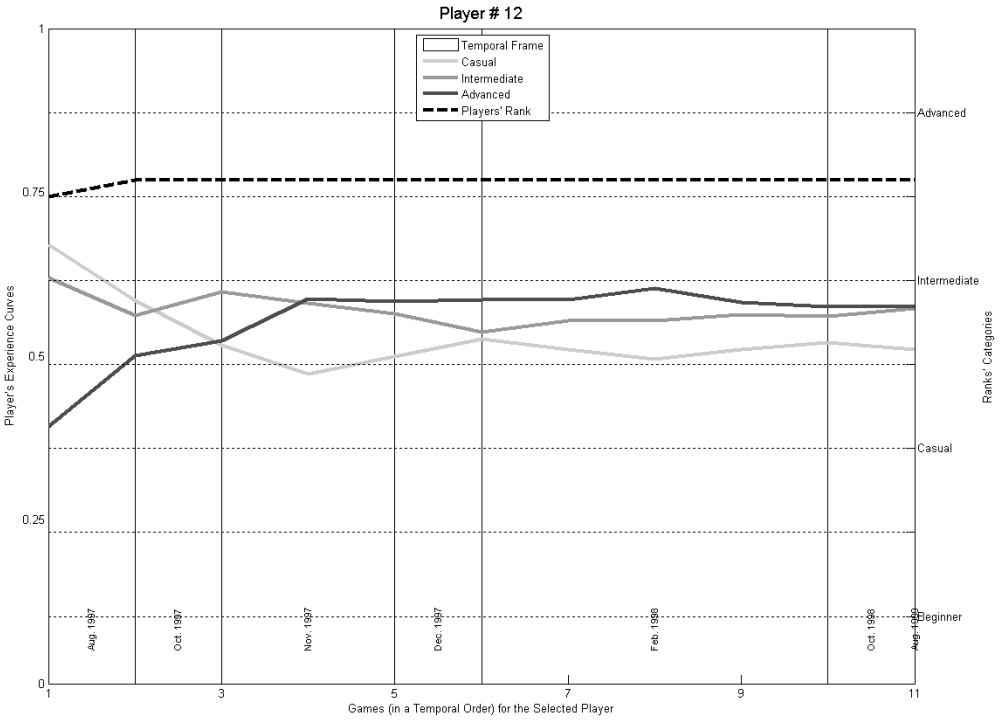


Figure 5.6 (l). The Competency-Level Monitoring Curve – Player #12; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

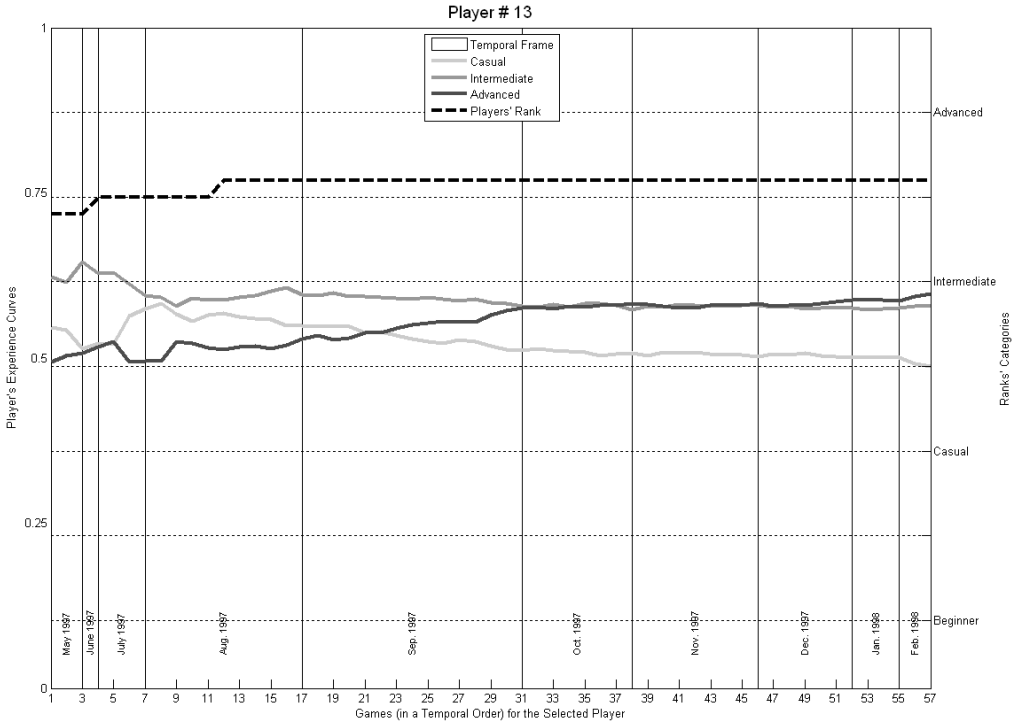


Figure 5.6 (m). The Competency-Level Monitoring Curve – Player #13; *Based on the situational reasoning of the individual player and the proposed three-tier classifier*.

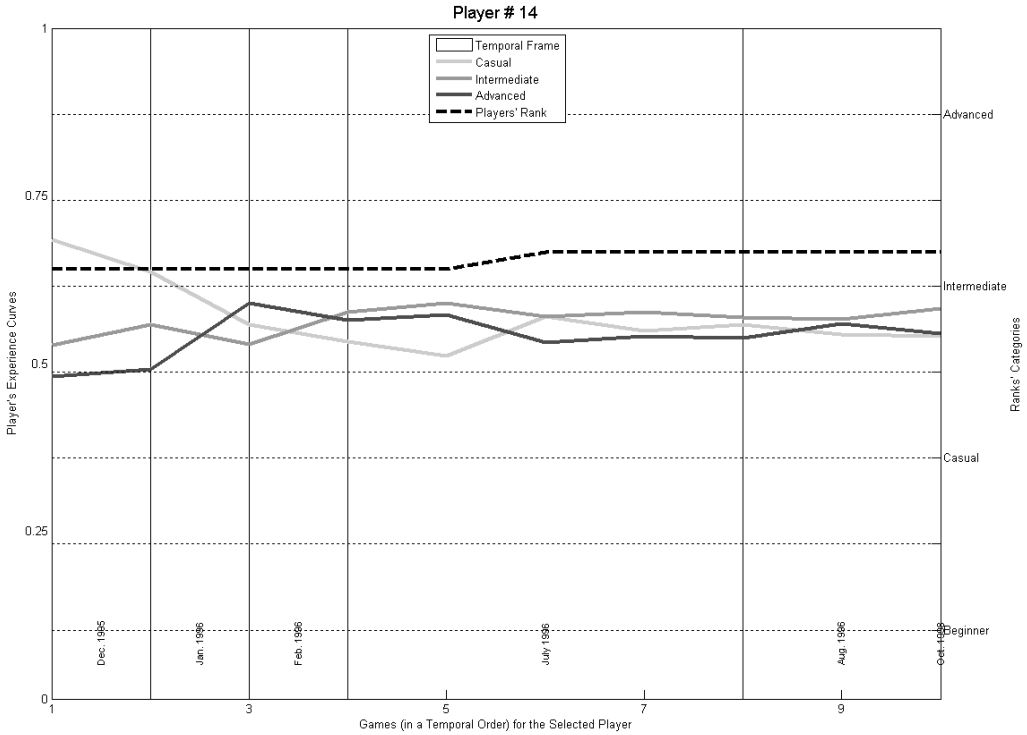


Figure 5.6 (n). The Competency-Level Monitoring Curve – Player #14; *Based on the situational reasoning of the individual player and the proposed three-tier classifier.*

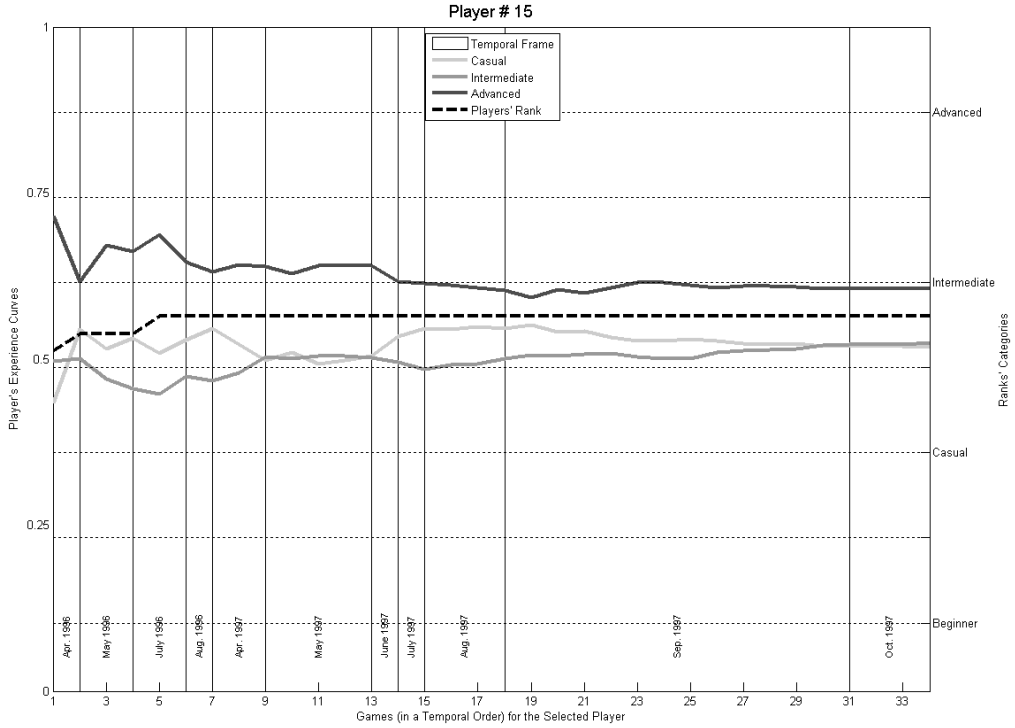


Figure 5.6 (o). The Competency-Level Monitoring Curve – Player #15; *Based on the situational reasoning of the individual player and the proposed three-tier classifier.*

Table 5.6 *(Continued in the next page).* The Correlation between the Selected Players' Ranks and the Competency Monitoring Curves *(Based on Situation Reasoning).*

Player ID	**Correlation Coefficient: *between the Experience Range and the …***			**The Averaged Experience Range Covered by the Corresponding Player's Games**	**Observations**
	Casual Curve	***Intermediate Curve***	***Advanced Curve***		
1	-0.57823	0.73832	0.24422	*Upper-Beginner to Lower-Intermediate*	A moderate negative correlation with the Casual curve accompanied by a Strong positive correlation with the Intermediate curve clearly indicates the acquisition of intermediate skills, and clearly reflects the expertise curve of the player.
2	0.54517	-0.6899	0.04977	*Lower-Intermediate to Mid-Intermediate*	The advancement within the intermediate range is not reflected by the positive moderate correlation with the Casual curve. However, it can be reflected by the moderate to strong negative correlation with the Intermediate curve.
3	-0.15879	0.49503	-0.43733	*Mid-Intermediate*	A moderate negative correlation with the Advanced curve and a moderate positive correlation with the Intermediate curve clearly indicate that the player is not improving.
4	-0.02841	0.62311	-0.15193	*Mid-Beginner to Mid-Casual*	The positive moderate correlation with Intermediate curves indicates the player's advancement towards the intermediate range.
5	No Variation in the Experience-Level, thus no trend is obtainable.			*Lower-Intermediate*	Not Applicable
6	-0.86608	0.86316	0.7959	*Mid-Casual to Lower-Intermediate*	The trend in the Experience Level is clearly shown by strong – respectively negative and positive – correlations with both Causal and Intermediate curves. A moderate to strong positive correlation with the Advanced curve exists.

7	-0.44773	-0.38765	0.79946	*Mid-Casual to Upper-Casual*	A moderate negative correlation with the Casual curve, accompanied with a low negative correlation with the Intermediate curve. A moderate to strong positive correlation with the Advanced curve exists.
8	-0.38135	0.1518	0.33005	*Mid-Casual to Upper-Casual*	A moderate negative correlation with Casual curve accompanied with a small correlation with the Intermediate curve reflect the expertise trend.
9	-0.53924	0.60584	-0.30341	*Upper-Intermediate*	A moderate – and weak, respectively – negative correlations with both Casual and Advanced curves, and a moderate positive correlation with the Intermediate curve.
10	-0.4239	0.07733	0.41045	*Lower-Intermediate to Mid-Intermediate*	The moderate negative correlation with the Casual curve and the moderate positive correlation with the Advanced curve are reflecting the experience trend.
11	No Variation in the Experience-Level, thus no trend is obtainable.			*Lower-Advanced*	Not Applicable
12	-0.86262	-0.72567	0.86867	*Lower-Advanced*	A strong positive correlation with the Advanced curve and strong negative correlations with both the Casual and Intermediate curves are clearly reflecting the experience level.
13	-0.40682	-0.72372	0.66887	*Upper-Intermediate to Lower-Advanced*	Moderate and high – respectively – negative correlations with both the Casual and Intermediate curves, accompanied by a moderate correlation with Advanced curve; clearly reflecting the experience level.
14	-0.32679	0.41054	0.0433	*Mid-Intermediate*	The weak negative correlation with the Casual curve and the moderate positive correlation with the Intermediate curve are reflecting the player's ranks.
15	0.49636	0.27864	-0.66489	*Lower-Intermediate to Mid-Intermediate*	The advancement within the intermediate range is somehow reflected by the weak positive correlation with the Intermediate curve. However, moderate correlations exist with both Casual and Advanced curves.

Table 5.6 (Continued). The Correlation between the Selected Players' Ranks and the Competency Monitoring Curves *(Based on Situation Reasoning)*.

5.4.4 Diagnosing Strategic-Learning

In the previous subsection, we have successfully demonstrated using an AC_R as an objective external-observer to estimate the strategic reasons behind the moves of human Go-players, those reasons were subsequently used in classifying the experience of the players and monitoring their strategic learning behaviour. In this subsection we reach the final stage of our results, in which we diagnose the learning activity of human Go-players by temporally observing each of the strategies' characteristics *(i.e., the aggregated subsets of reasons)*. A benefit of the following figures is the distinctive opportunity to realize how the strategic reasoning of human Go-players is decomposed among the available strategies' characteristics, and how those characteristics evolve temporally with experience. The figures below show the un-weighted Cumulative Moving Average (CMA) — again with a maximum window size of *50* games — of the Reasons Frequencies per Step (FS), as directly measured from the games. The FS calculated only from the reasons behind the — corresponding — selected player's moves. To allow mutual comparisons, each figure is normalized by the corresponding highest averaged characteristic-value.

As we have mentioned, the five finally-selected players were found to cover the whole available experience range. In fact, they tend to cover the whole experience range in somewhat mutually-exclusive sub-ranges. Therefore, a possible scenario to visualize and hypothesize-about the overall strategic-learning process is by linking them successively with a simple underlying storyline. As a Go-player progresses from being a Beginner to lower-Intermediate—through being a Casual—the categories 'All Reasons', 'Considered an Attack', 'Considered a Defence', and 'Thoughtful' seem—in general—to decline slightly with experience. The 'Explicit Gains' appears to be the only curve that rises during the Beginner to lower-Intermediate progression. On the contrary, progressing from the Intermediate to Advanced shows precisely the opposite behaviour. As the player progresses through the Intermediate rank towards the Advanced rank, the characteristics 'All Reasons', 'Considered an Attack', 'Considered a Defence', and 'Thoughtful' unevenly increase with experience. Unsurprisingly, the 'Explicit Gains' is

the only curve that declines during the Intermediate to Advanced progression. However, both categories the 'Not Recommended' and the 'End of the Game' show very subtle variations throughout the entire experience range.

In Subsection 5.3.4, and in order to investigate the differences in the Go-strategies' characteristics, statistical analysis methods were applied to measurements from the mid-range games of Casual, Intermediate, and Advanced expertise. The findings, then, implied that as a player progresses from Casual to Advanced, the strategies employed become more thoughtful, with a tendency to attack rather than being defensive, and of also being less attracted by instant (explicit) gains. These earlier findings seem to agree with only half of the hypothesized storyline, that is, the changes occurring as a player progresses through the Intermediate rank and to being an Advanced. This apparent disagreement—where the categories 'All Reasons', 'Considered an Attack', 'Considered a Defence', and 'Thoughtful' seem to decline as a player advances from being a Beginner to a lower-Intermediate—can be attributed to two associated reasons. To begin with, in Subsection 5.3.4, no Beginner games matched the selection criteria, and therefore, the progression from Beginner to Casual was not investigated. That leads us to the second reason; the window size of *50* games employed in the CMA considers this 'history' of being a Beginner when the player has already advanced onto being a Casual, thus affecting the curves for an additional period—as in both players number *1* and *4*.

An alternative — yet vital — approach is to read the results by considering each figure as a way to monitor a personalized strategic-learning activity. Each figure characterizes how the corresponding player is evolving with the experience he/she is gaining. The plain benefits of such results, is the ability to construct customized — whether to a specific personality or to a level of expertise — learning processes, consisting of designed tasks that convey to a learner the missing bits of knowledge/skill/understanding, by which gaining experience might be assured and/or accelerated. Another potential benefit is the ability to clone a person/level-of-expertise, possibly for constructing an automated instructor.

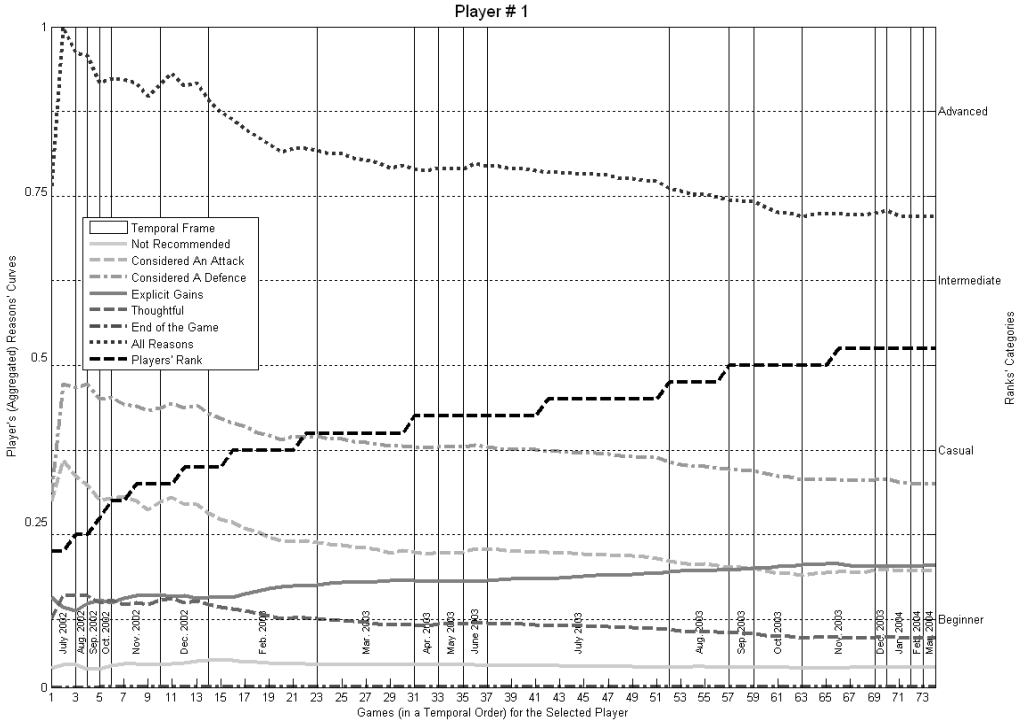


Figure 5.7 (a). The Skills – *Reasoning Diagnosis* – Monitoring Curves – Player #1.

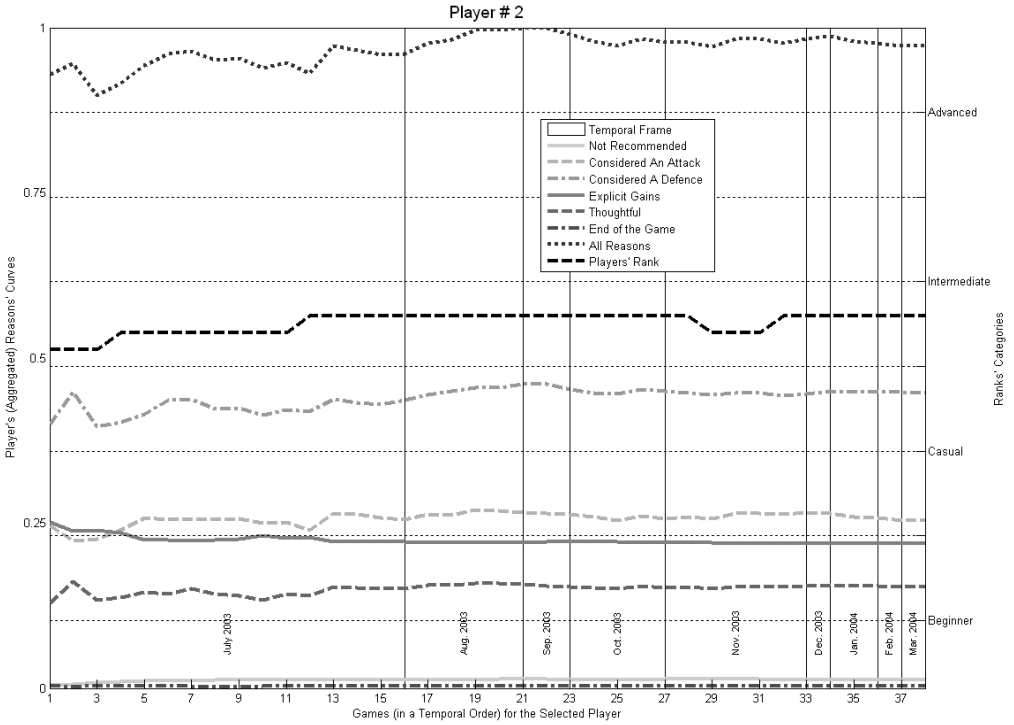


Figure 5.7 (b). The Skills – *Reasoning Diagnosis* – Monitoring Curves – Player #2.

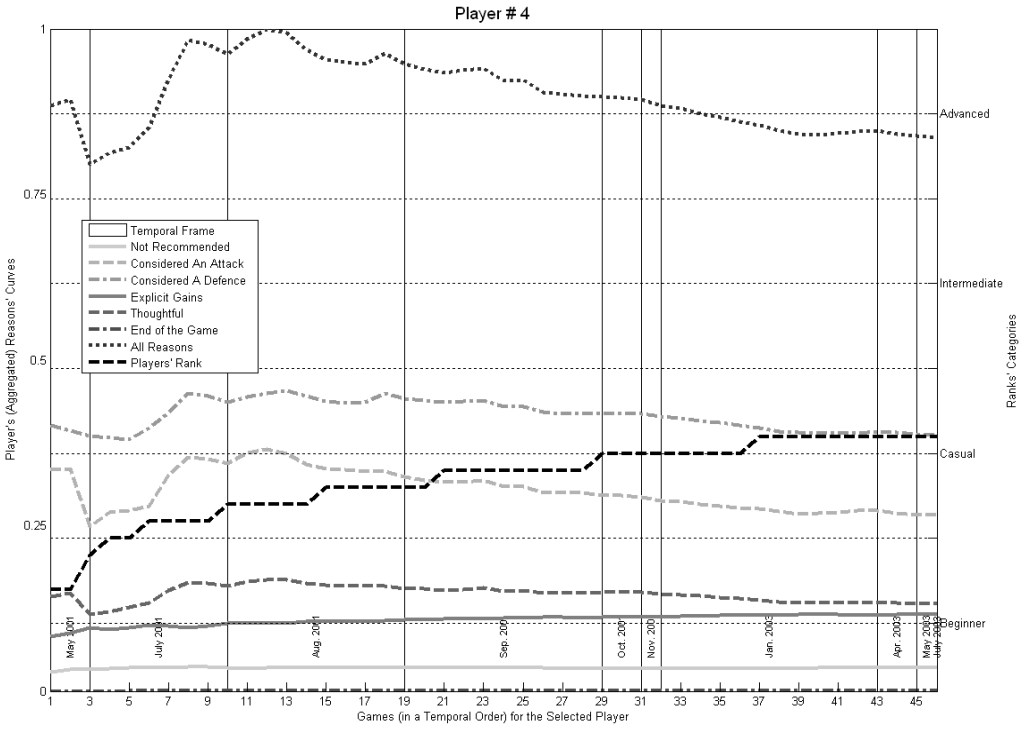


Figure 5.7 (c). The Skills – *Reasoning Diagnosis* – Monitoring Curves – Player #4.

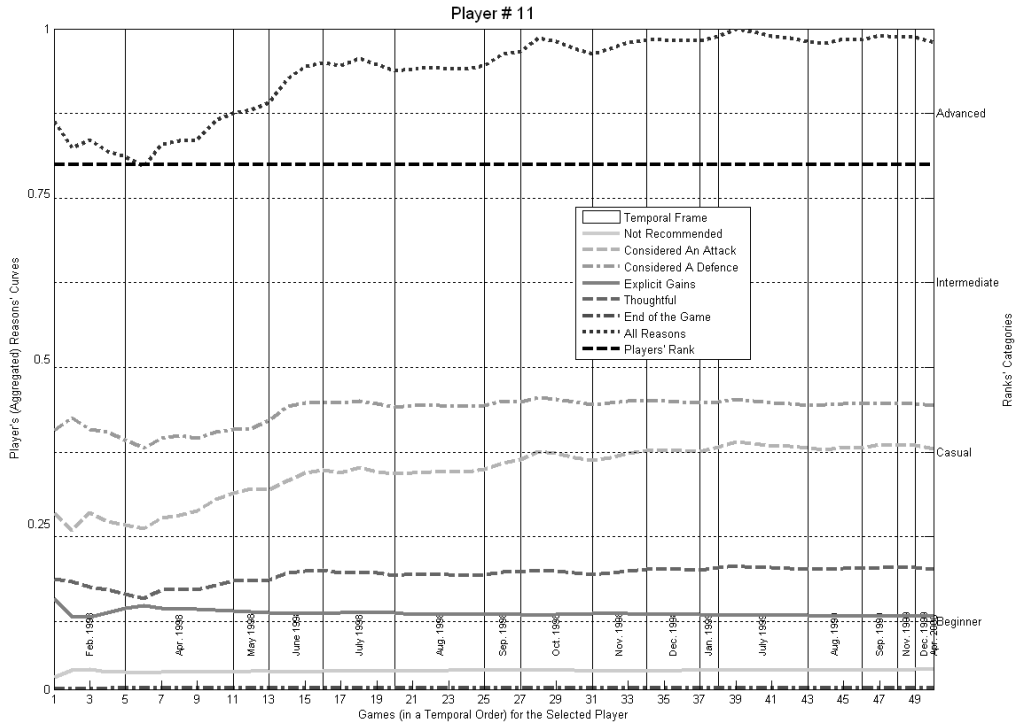


Figure 5.7 (d). The Skills – *Reasoning Diagnosis* – Monitoring Curves – Player #11.

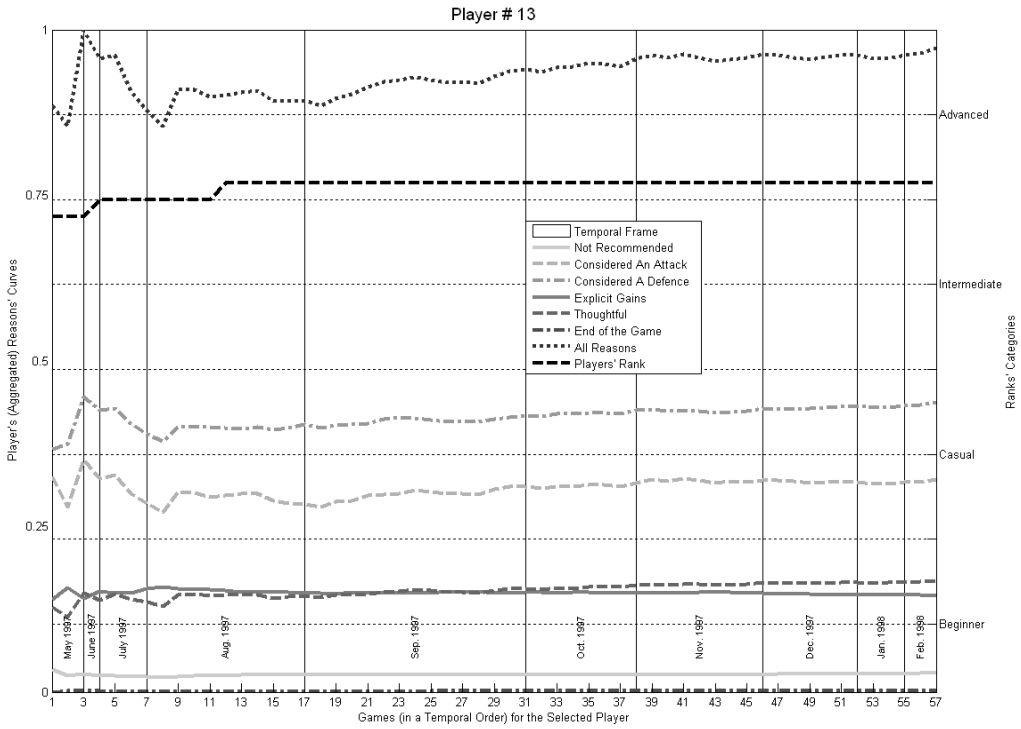


Figure 5.7 (e). The Skills – *Reasoning Diagnosis* – Monitoring Curves – Player #13.

5.4.5 Effect of Changing the Classifiers' Architecture

Although the classifier's second-tier — composed of a forest of *30* RFs — is not based upon RFs that are trained using randomly-drawn subsets of the training dataset, it can be considered a form of bagging in the sense that it is in fact composed of RFs with minor changes in their operational parameters. That is, those RFs are varying in the number of classification trees they consist of, in addition to the randomness existing in the constituting trees due to both operators — i.e., bagging and random feature selection — used while generating each single RF.

In order to investigate if combining those RFs would lead to an improved overall performance, we will only select the single best RF and re-run the whole experiment, instead of using all the *30* RFs as previously demonstrated. The resulting figures of the finally-selected five players are shown below.

In general, the figures show that combining the relatively weaker RFs to form an ensemble indeed created the hypothesized improved performance. In "Player #1", besides a misclassification during the very early *2* games, the last dozen games are misclassified as Advanced rather than Casual as before. While both classifications are — in a very strict sense — incorrect, Casual is more appropriate considering that the actual ranks are a 'lower' Intermediate. The lower performance can also be demonstrated in "Player #2", where the entire Advanced curve is erroneously higher than the Intermediate curve. Also, Player #4 is correctly classified as a Casual, yet the monitoring curves show an abrupt rising of the Advanced curve above the Intermediate. Similar relatively-abrupt changes can be seen in the beginning games of both player number *11* and *13*, although — in both players — the rest of the temporal frames are correctly classified. In fact, the monitoring curves of the "Player #13" show an improved classification, in which the Intermediate curve falls below the Advanced curve starting from the game number *18*, instead of *34*, as in the previous results.

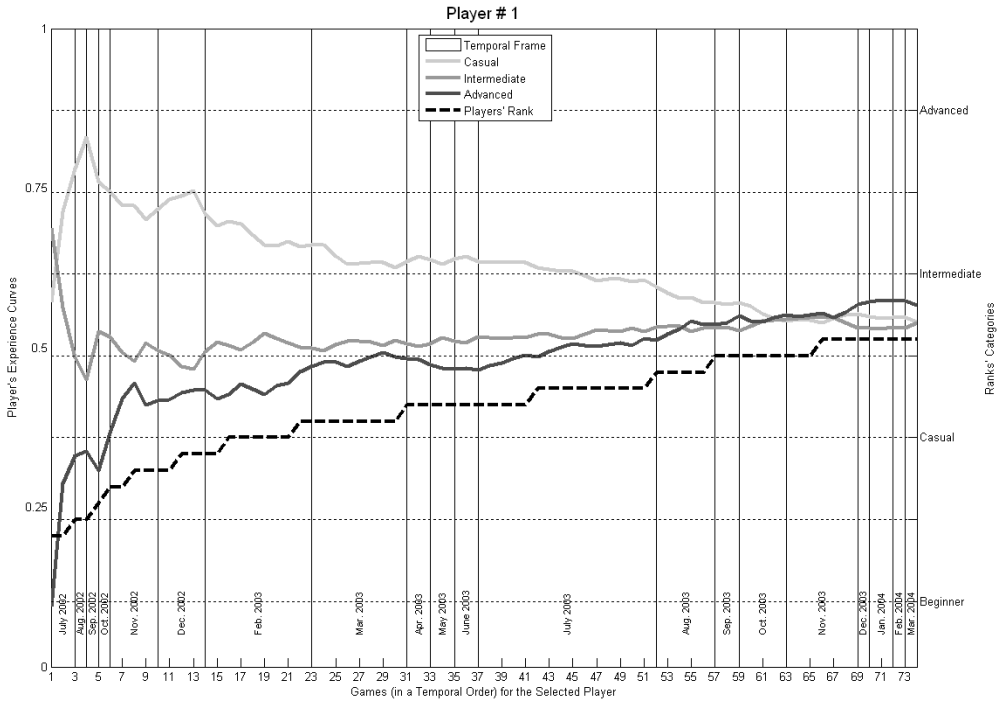


Figure 5.8 (a). The Competency-Level Monitoring Curve – Player #1; *Based on the situational reasoning of both opponents and an alternative architecture for the classifier*.

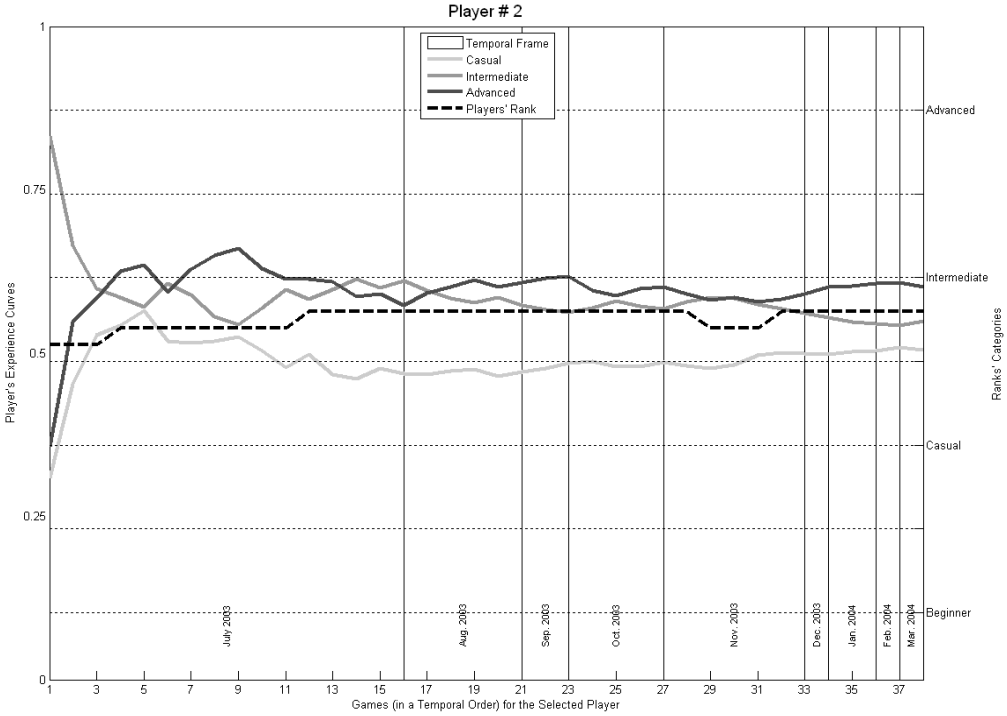


Figure 5.8 (b). The Competency-Level Monitoring Curve – Player #2; *Based on the situational reasoning of both opponents and an alternative architecture for the classifier*.

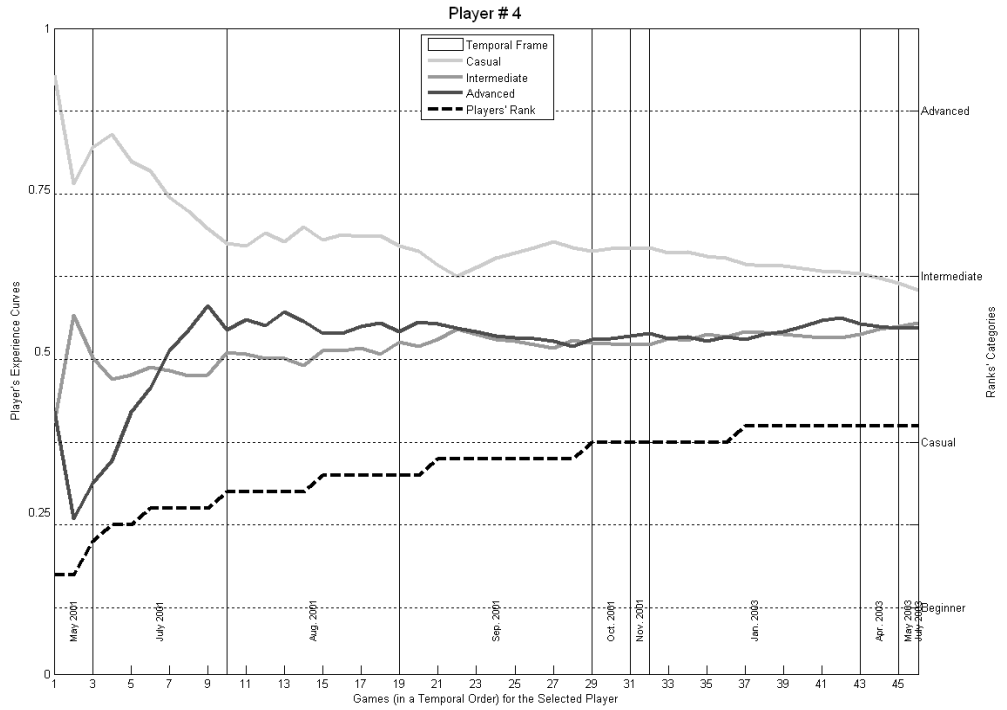


Figure 5.8 (c). The Competency-Level Monitoring Curve – Player #4; *Based on the situational reasoning of both opponents and an alternative architecture for the classifier*.

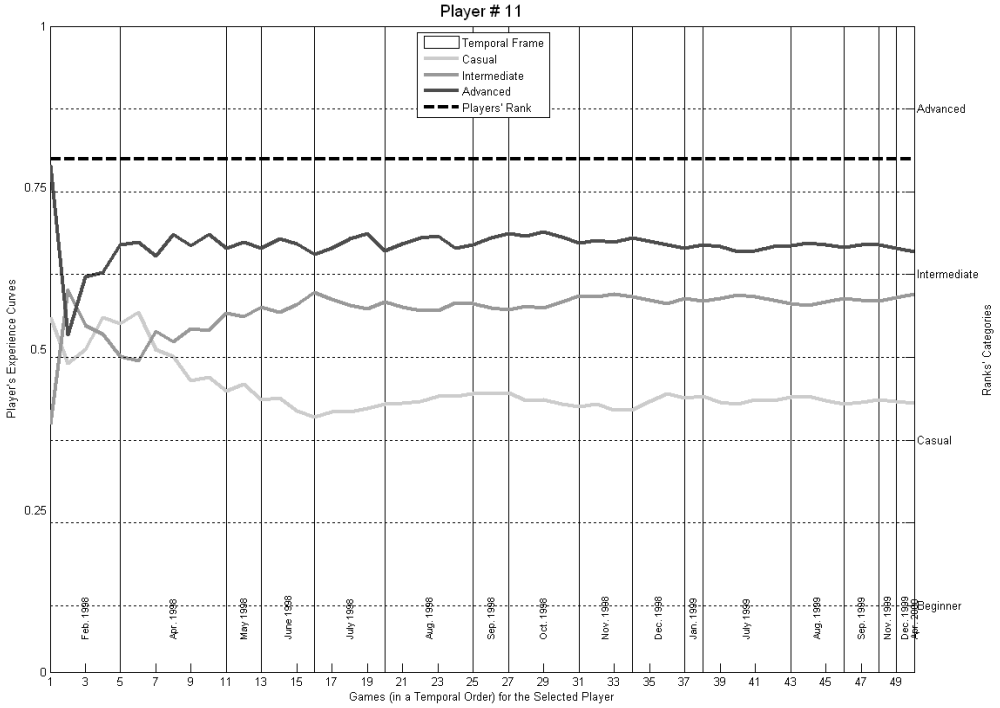


Figure 5.8 (d). The Competency-Level Monitoring Curve – Player #11; *Based on the situational reasoning of both opponents and an alternative architecture for the classifier*.

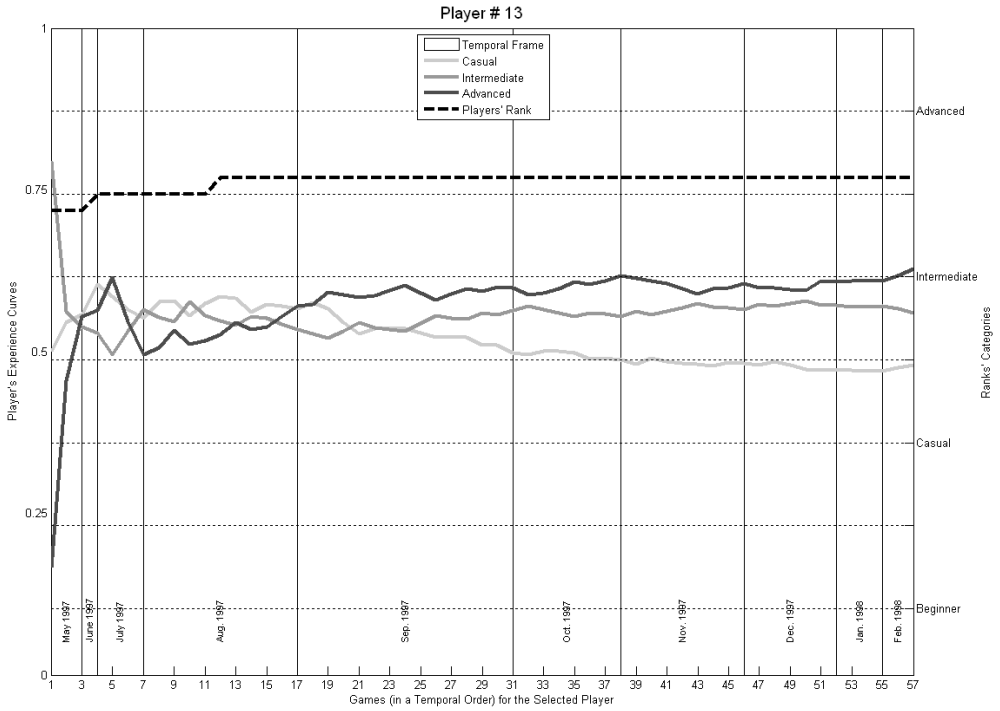


Figure 5.8 (e). The Competency-Level Monitoring Curve – Player #13; *Based on the situational reasoning of both opponents and an alternative architecture for the classifier*.

5.5 Chapter Summary

In this chapter, a methodology was provided for an automatic and objective discovery of the properties of the strategies employed by human-players' in the game of Go, and explored the changes in the strategies as the players gain experience. We view this work as a step towards creating an explanatory link between the human mind and machine, and of reproducing human intelligence.

In more details, the previously presented computational framework was adapted, so that the connectivity-patterns assessment component that provided an evaluation in terms of network motifs was replaced by a reasoning assessment component that provided declarative reasons as explanations, to the moves played by human Go players. That component have been realised using a publicly available engine which is capable of providing the required explanations. The explanations covered the key concepts of reasoning within the Go domain; namely, the concepts of attacking, defending, looking-ahead, and capturing. The engine was successful in analysing a training dataset that is composed of hundreds of games suitably selected from an online server, and the results demonstrated a statistically-significant capability in characterizing varying levels of expertise in Go.

As in Chapter 5, statistical measures were applied to the results of this analysis, in order to generate features upon which a competency model was developed. The model, developed as a three-tier ensemble-classifier based on random forests, was more-successful in predicting the class label of a game of Go when compared to the connectivity-patterns based classifier. This capability was demonstrated using a test dataset of *15* human players, whom which the proposed approach successfully provided a temporal-monitoring for their overall expertise-level along with their specific skills-levels. In addition, this chapter investigated alternative factors within the approach's operational framework, in which *analysing individual players rather than games* and *a modified architecture for the proposed classifier* were examined.

The results represent a distinctive opportunity to realize and monitor how the strategic reasoning of human players is decomposed among the available strategic characteristics, and how those characteristics evolve temporally with experience. The findings are seen as advancement towards a more customized learning processes, and as an innovative tool in the field of computational psychology for the modelling—and potentially the imitation/reproducing—of human behaviour.

[This Page is Intentionally Left Blank]

Chapter 6: Semantic Evolution of Neuro Go-Players

The work, reported in this chapter, has been partially published in following articles:

Amr S. Ghoneim, and Daryl L. Essam (2012). A Methodology for Revealing and Monitoring the Strategies Played by Neural Networks in Mind Games. Accepted by the 2012 International Joint Conference on Neural Networks (IJCNN), at the 2012 IEEE World Congress on Computational Intelligence (WCCI 2012), Brisbane – Australia, June 10-15.

Amr S. Ghoneim, Daryl L. Essam, and Hussein A. Abbass (2011). On Computations and Strategies for Real and Artificial Systems. In Advances in Artificial Life, ECAL 2011: Proceedings of the Eleventh European Conference on the Synthesis and Simulation of Living Systems, edited by Tom Lenaerts, Mario Giacobini, Hugues Bersini, Paul Bourgine, Marco Dorigo and René Doursat, pp. 260-267, 8-12 August 2011, Paris, France. MIT Press.

In Chapter 4, an automated competency assessment environment was proposed, and developed, to monitor human Go players on 19×19 boards. In Chapter 5, through utilizing situation reasoning, the proposed environment achieved its objective by providing an effective competency-monitoring of the human Go players. However, to effectively support the development of computer Go players, two issues should be addressed.

To begin with, the proposed methodology should be used to assess computer Go players, and to be capable of assessing their performance on smaller board sizes. As we have mentioned, smaller board sizes – *which are less complex than the 19×19 board, yet complex enough to represent a challenge for novice human players and computer Go* – are widely exploited, in particular the 9×9 variant. However, a measure of competency is not effective without consequent actions. Thus, in order to effectively support the development of computer Go players, the second issue is to actually utilize the methodology to effectively guide and train selected computer Go agents. In this chapter we address both issues, which correspond to the third and fourth sub-questions (see Section 1.3).

This chapter starts by extending our proposed framework – *and our findings in Chapter 5* – to target one of the disadvantages in neuro-evolving Go Players; That is, by semantically unfolding the evolution of Go neuro-players, and monitoring the overall dynamics of the evolution process. Thus it will be possible to observe any stagnation in the evolutionary processes, and to extracting the semantics of a neural networks' behaviour.

Then, later in this chapter, we address the issue of recommending a follow-up action that would support an improved development of computer Go agents, thus practically addressing the fourth sub-question within this study (see Section 1.3). That is, to investigate the value an automated competency assessment, leading to an enhanced awareness, would bring to the advancement of computer Go agents. The hypothesis is that improved competency assessment will positively affect computer Go, by driving the development of better Go-playing capabilities in the agents. Thus, this chapter proposes extending our framework to develop a strategically-aware fitness function to guide the neuro-evolution, and investigates how this proposed function would affect – *in comparison to the traditional fitness function* – the development of improved competencies within the evolved Go agents.

The first part of this chapter starts with a timely review – *in Section 6.1* – of the challenges to the application of neuro-evolution within the Go domain, and thus the need to unfold the dynamics of the evolutionary process. Section 6.2 presents the approach proposed to semantically monitor the neuro-evolution of Go players, including an experimental setup. Section 6.3 follows by a discussion of the results. The second part of this chapter starts in Section 6.4 by overviewing the background knowledge concerning the incorporation of domain knowledge into the evolutionary process, and its anticipated rewards on the neuro-evolution of Go players, followed by describing the proposed fitness function and the experimental setup in Section 6.5. A comparative study of both fitness functions – *the proposed versus the traditional* – is presented in Section 6.6. Finally, the chapter and its findings are summarized in Section 6.7.

6.1 The Need to Unfold the Neuro-Evolution

To become different from what we are, we must have some awareness of what we are.
–––Eric Hoffer (1902 – 1983)

Computational Intelligence (CI) has been successfully applied to various applications, including mind-games, where its intention is to achieve high-quality human-like game playing capabilities. Due to their ability to learn autonomously and to effectively recognize and transform patterns, Neural Networks have been successfully applied to various mind-games (Mańdziuk, 2007). Neuro-Evolution is an approach to train neural networks using an evolutionary algorithm (Richards et al., 1998). For it, the evolutionary approach searches for the network parameters that maximize its performance in learning a specific task. However, despite its advantages when compared to the traditional 'supervised' training methods, neuro-evolution may suffer from drawbacks; for instance, evolutionary convergence to a local optima that is widely known as stagnation. From the previous review on Machine-Learning approaches to computer Go, and in particular neuro-evolution (see Section 2.2.4), the proposed methodologies dealt with issues that inherently revolved around two main concerns:

- The nature of the game-playing strategies evolved *(What type of strategies are evolved?)*, and
- The dynamics of the game-playing strategies evolution *(Are there any strategies evolving?)*.

Neuro-evolving against an opponent leads to types of strategies that are limited in quality to the strength of that opponent. Evolving against varying opponents is fundamentally about evolving varying types of strategies to achieve good play, and is based on figuring out how much diversity in the types of strategies is needed to achieve good play. Co-evolution, and the subsequent cultural co-evolution, attempts to eliminate this limitation to the nature of the strategies evolved; as well as assuring a continuous

progress, without stagnation. When scalability was the concern, it was about evolving the right type of strategies that could be valuable on a different scale. To sum up, in order to realize the full potential of neuro-evolution in games, unfolding the evolution of neural game players is important, if not mandatory.

Stagnation is superficially detected when evolution fails to improve on existing solutions, as measured by its fitness function. In mind-games, this in effect corresponds to a player that fails to improve its skills – thus its performance remains constant. However, if during evolution we see a plateau in the fitness function, it does not necessarily mean that evolution has stagnated. Sometimes it takes time to re-construct building blocks, where this area of perceived stagnation is followed by a sudden jump in fitness. Similarly, a continuous increase in fitness does not mean that the neuro-players are improving their skills. Sometimes they exploit a weak player or niche, where the continuous increase in fitness is a false indicator of any improvements.

Therefore, one can imagine that the use of a proper mechanism to assess the skill of an evolving player can provide one way to reveal evolutionary dynamics. That is, the types of strategies evolved within a neural network can be measured, and thus the dynamics of the neuro-evolution process itself can be monitored. This challenging issue is traditionally answered through rule-extraction techniques (Andrews et al., 1995). However, due to the compactness of NNs, the acquired knowledge is represented numerically and never explicitly declared. This numerical representation, besides the architecture and the weighted connections, affects the expressive power of the extracted rules. Most rule-extraction methodologies express the mined rules in the conventional Boolean/Propositional Logic, or in the less usual Fuzzy Logic with membership functions. In both cases, the *if ... then ... else* form is widespread. We call this representation a syntactical representation of the network. The reason for this, is that it captures the internal representation as a function of the neurons' activities, without necessarily representing the meaning of the behaviour of the network as a whole. The latter is what is known as a strategy within a mind-game.

In other words, due to the nature of NNs and of the extraction techniques, the extracted rules do not have the necessary expressiveness to explain game-playing strategies. However, in practice, the neuro-evolution approaches need to discover the semantics of the network's behaviour; the general strategies and reasoning that maps the goal(s) of the game into actions. Therefore, the conventional 'Boolean' representation lacks the expressiveness needed to describe such strategies.

The first question addressed by this chapter, is of how to describe the types of strategies that evolve in a neural game-player across a given number of generations. As well as of how to detect whether the evolved networks are improving towards the required objective, or if the evolution has stagnated, and thus if the networks are not developing new strategies. Thus, we present an attempt to answer these questions by extending the role of the proposed competency assessment approach to provide domain-specific semantics to the moves played by the evolved neural-players. It is hypothesized that the selected situation reasoning is capable of addressing these questions by correlating with – and consequently, assessing – the performance and competency of the evolved neuro-Go players.

6.2 Proposed Methodology: Semantically Monitoring the Neuro-Evolution of Computer Go Players

Our proposition (Figure 6.1), is that the established computational framework (see Chapter 5) can unfold the evolution of a NN player by providing the strategies and reasoning behind the moves played by the network. The explanatory/analysing component can replay a game, providing an explanation from its own perspective – *i.e., depending on its internal structure and dependencies* – to the moves played in that game. Thus, by replaying the games played by a NN at different stages – *generations* – of an evolution, we can understand the dynamics of the evolution; whether the networks are evolving any strategies or are idle. We can also discover and estimate the types of game-playing strategies evolved. In addition, the generated explanations can be

combined with a data-mining technique to provide the probability for the evolution process to converge, and consequently to potentially notice stagnation.

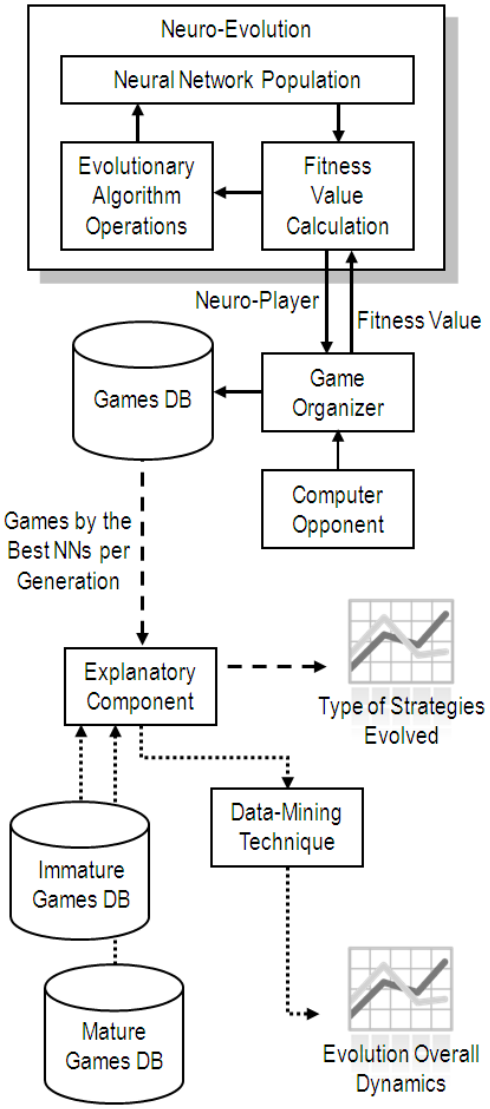


Figure 6.1. Conceptual Diagram of the Proposed Methodology. Solid Arrows represent the Neuro-Evolution Process of a Game-Player, Dashed Arrows represent the Process of Analysing the Type of Strategies Evolved, while the Dotted Arrows represent the Process of Analysing the Evolution's Overall Dynamics.

To realize our proposition, the Go neuro-players were evolved using SANE; a methodology which has been repeatedly utilized within the Machine-Learning approaches to computer Go (Richards et al., 1998). GNU Go (Bump et al., 2009) was selected as the analysing engine *(as in the preceding "Chapter 5" settings)*, as well as the opponent in the evolution process. Finally, Random Forests (Breiman, 2001) were selected – *also as in the previous setting* – as the data-mining technique. Details of the SANE methodology and the experimental setup are presented in the following section.

6.2.1 Symbiotic Adaptive Neuro-Evolution

SANE (Moriarty and Miikkulainen, 1997) is a different approach to neuro-evolution. Instead of evolving a complete network as a potential solution, two separate populations are evolved simultaneously, network blueprints and neurons. The neurons act as local solutions that explicitly decompose the search space, while the blueprints search for a solution to the best combinations of neurons *(instead of building the networks out of randomly selected neurons)*. This methodology was also applied to another mind game, Othello (Moriarty and Miikkulainen, 1995). SANE traditionally evolves neurons for a single hidden layer, thus in fact evolving a three-tier feed-forward network.

Each board intersection has two input nodes *(one for each player)*, and a single output node. For an intersection, the first representative input node is activated iff the intersection is occupied by a white stone, and vice versa if the intersection is occupied by a black stone, the second input node is activated. Therefore, an empty intersection is indicated by deactivating both input nodes, and it is illegal to activate both. The output node has a sigmoid activation function; the next move is represented by the highest value *(a larger value corresponds to a better move)*. If all values are below 0.5, the network passes. If the selected move is illegal, the move corresponding to the next highest activation is selected. Each neuron defines a fixed number of weighted connections that are randomly assigned to both input- and output-layer nodes.

In the evaluation phase, a network plays a game of Go against an opponent, the fitness value is merely the game's final score. As for each neuron, its fitness value is the normalized summation of the fitness values of the networks in which it participated. Single point crossover is then applied on mates selected from the elite one third of the networks, and 25% of the neurons, thereby creating two offspring that replace the worst individuals. Mutation is then applied conservatively to the neuron population, and more aggressively to the blueprints *(to maintain high diversity among the network)*. Table 6.1 clarifies the application of SANE to evolve neuro Go players.

6.2.2 Experimental Setup

SANE is used to evolve networks playing on *9×9*, which is the current limit in the neuro-evolution literature. Table 6.1 highlights the network, and the evolution parameters. GNU Go version *3.6* is used throughout the experiments. The engine's level was set to *1*, the default is *10*. Chinese rules were used to score the Go games. The networks are always evolved to play White, thus never making the first move. The *komi* value – which is necessary to avoid a tie – was set to *0.5*, and no handicap stones were given to the networks. An upper bound of *200* moves per game was placed, to ensure that unreasonably long move-sequences that are probably suggested by the untrained networks are not pursued.

The actual score of the game is used as the fitness value. There is not a separate output value for *pass*, thus a network passes whenever the highest output value is less than a threshold – *0.5* in this study – signalling that no good moves can be found. In general, the parameters in the experiments are based on those found effective in (Richards et al., 1998). A single run consists of *350* generations, and *10* different runs were evolved. The *350* generations are about *100* generations more than that required by SANE to evolve a network capable of defeating Wally (Richards et al., 1998). However, GNU Go – *even when playing at level 1* – is much more developed than Wally. Therefore, we do not expect to evolve a NN that is capable of defeating GNU Go, but rather a network that develops enough strategies to explore.

Table 6.1. Pseudo-code of SANE Applied to Evolving Neuro-GO Players.

STEP 1: Generate Population
Generate an initial random population P_n of *4000* neurons
… per neuron: *12* weighted connections
Generate an initial random population P_b of *200* blueprints
… per blueprint: *162* input nodes, *81* output nodes, *500* hidden nodes

For each generation *gen* of *350* generations
STEP 2: Evaluate Population
For each neuron *n* in population P_n
n.fitness ← *0*
n.participation ← *0*
For each blueprint *neuralnet* in population P_b
neuralnet.fitness ← **game-score**(*neuralnet* vs. *GNU Go*)
For each neuron *n* in *neuralnet*
n.fitness ← *n.fitness* + *neuralnet.fitness*
n.participation ← *n.participation* + *1*
For each neuron *n* in population P_n
n.fitness ← *n.fitness* / *n.participation*

STEP 3: Sort Population and Select Mating Individuals
sort(P_n , Quick-Sort)
$Elite_n$ ← Best *1000* neurons that are allowed to breed
sort(P_b , Insertion-Sort)
$Elite_b$ ← Best *67* networks that are allowed to breed

STEP 4: Apply Evolutionary Operators
crossover(P_n , $Elite_n$, One-Point-Crossover)
mutate(P_n , Mutation-Rate: *0.1*)
crossover(P_b , One-Point-Crossover)
mutate(P_b , $Elite_b$, Mutation-Rate: *0.1*)

As in the prior setup (Chapter 5), the generated reasons are grouped into a small number of sets (Table 5.1) according to tactical/strategical similarities to facilitate identifying the general strategies and observing any potential trend; those groups are: 'Not Recommended', 'Offensive', 'Defensive', 'Direct/Explicit Gains', 'Thoughtful', 'End-of-Game', and 'All Reasons'. The main feature that was then calculated from the reasons list per game was the following measurement:

- The *Reasons Frequencies per Move* (Eq. 5.2) which correspond to the frequency of each reason per game divided by the total number of moves in the corresponding game.

This feature is the input to the Random Forest (RF) – *which is once more selected as the data mining technique* – that was trained to classify a game as either immature (i.e., untrained) or mature (i.e., fully trained). The RF is trained using two databases (Figure 6.1) of features, each containing *150* profiles. The profiles of the immature games are created from *150* games played by the networks evolved in the very first *3* generations, thus virtually having no game-playing capabilities. As for the mature game, the profiles are created from games played by a GNU Go tournament; since GNU Go is the only source of information the networks have, the strategies found in GNU Go played games can be considered the highest level the networks are trying to achieve. The RF is then trained to generate a probability of being immature or not. In all the forthcoming figures, a cumulative moving average spanning *50* generations is used.

6.3 Results and Discussion

We have run our system using SANE for *350* generations. Figure 6.2 shows the average score of the game – *which cannot go below -80.5 given our settings* – as the networks evolve. The average of the *10* runs and the *200* networks show a clear improvement for the first *100* generations, after which both curves enter "an almost flat" plateau. An initial and very simple demonstration of the usefulness of using generated explanatory reasoning is to track the evolution as shown in Figure 6.3. As the networks evolves, the number of reasonless moves drops from an average of *24* per game to

around *19*. That is a clear trend that indicates the development of some game-playing strategies/reasoning with evolution.

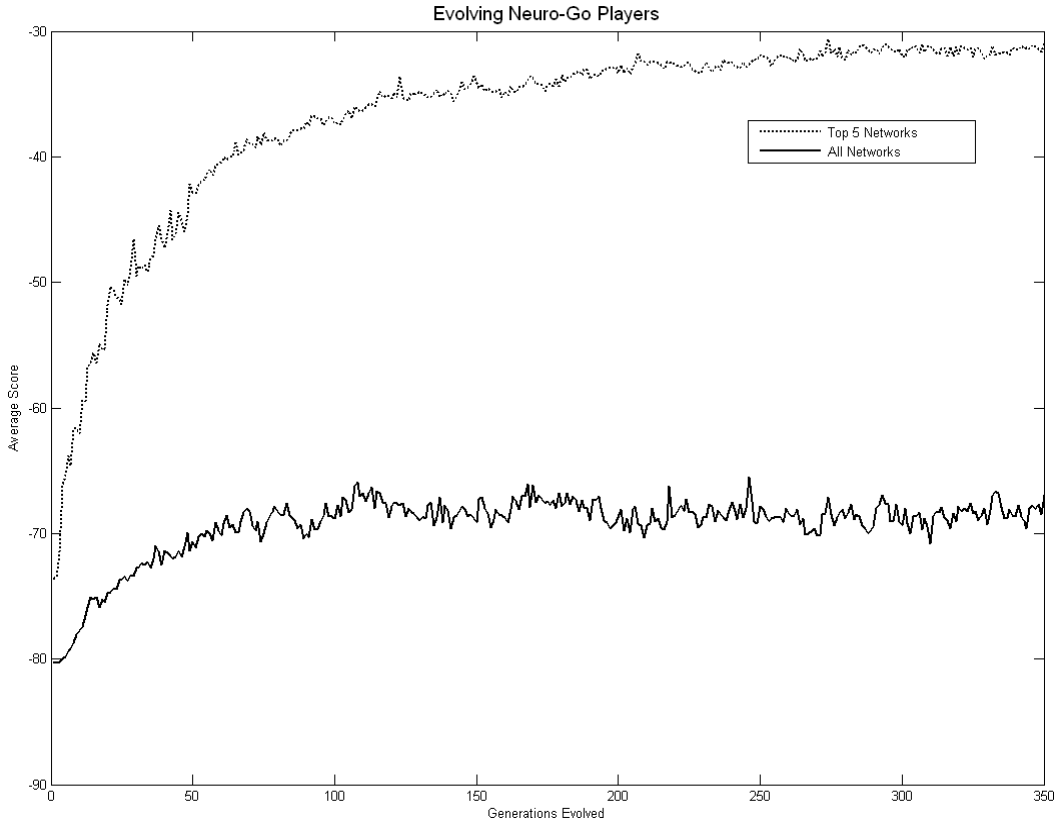


Figure 6.2. Evolving Neuro-Go Players.

6.3.1 Types of Strategies Evolved

To investigate the types of the strategies evolved, Figure 6.4 shows the curves for the aggregated sets of reasons. Although, practically, the results sharply vary on an individual level, the smoothed curves show a trend for the first *100* to *150* generations. The average moves with a *Not Recommended* strategy decrease by almost *2* per game, and most importantly, the average number of moves with a *Direct/Explicit Gain* strategy rises from *1* to about *4* per game. The average moves with an *Offensive*, *Thoughtful* and *End-of-Game* strategies did not vary significantly with evolution. All these findings were expected. Being *Thoughtful*, *Offensive* rather than *Defensive*, and aware of the *End-of-Game* patterns are practiced by relatively more mature players. In

addition, avoiding some *Not Recommended* moves, and – *more importantly* – being attracted by *Direct/Explicit Gains* are commonly the strategies adopted by untrained players, besides being more *Defensive*. Therefore, finding that *Defensive* moves decreased by almost *2* moves per game was interesting. The reason being that the *Defensive* moves – *rather than being lost to Offensives* – are being misplayed instead to capture a stone or to expand its territory – *which are part of the Direct/Explicit Gain moves* – by the immature networks. Thus, suggesting that being *Defensive* is a capability that is yet to be developed in the players.

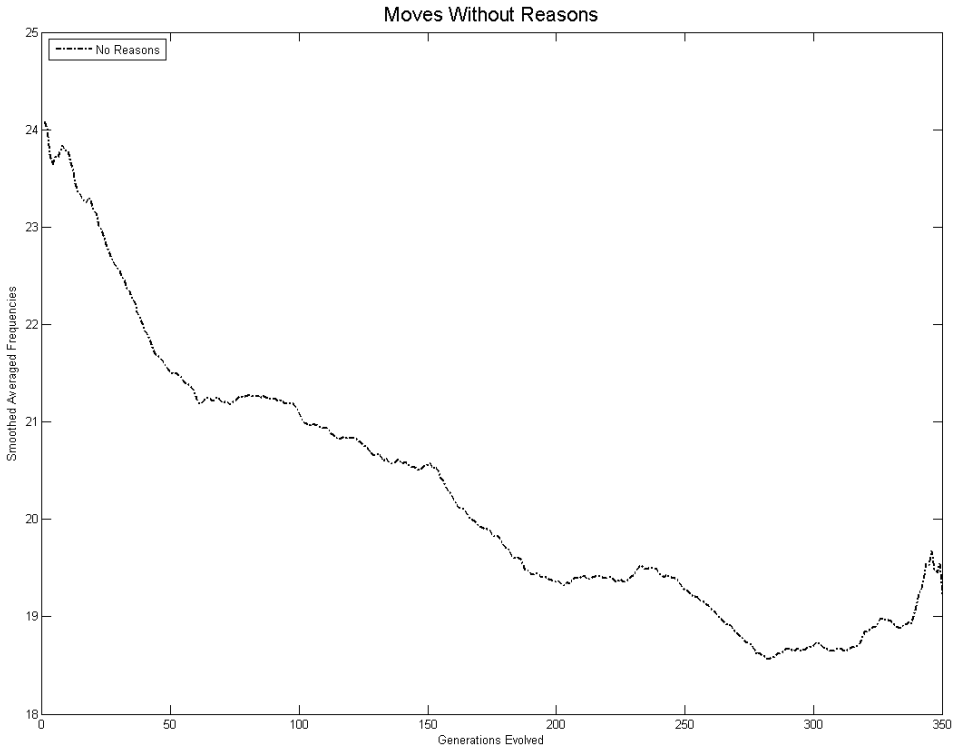


Figure 6.3. Number of "White" Moves without any Reasons per Game.

Figure 6.5 shows a more detailed – *close-up* – on the *Direct/Explicit Gain* strategy which showed the largest variation – *increase* – in terms of moves. The curves that represent each constituent reasoning in that category are displayed. It is obvious from that figure, that the networks quickly evolve the "expand territory" strategy. However, these expansion attempts were done without any real increase to its "expands moyo", "is an invasion" or "captures something" strategies.

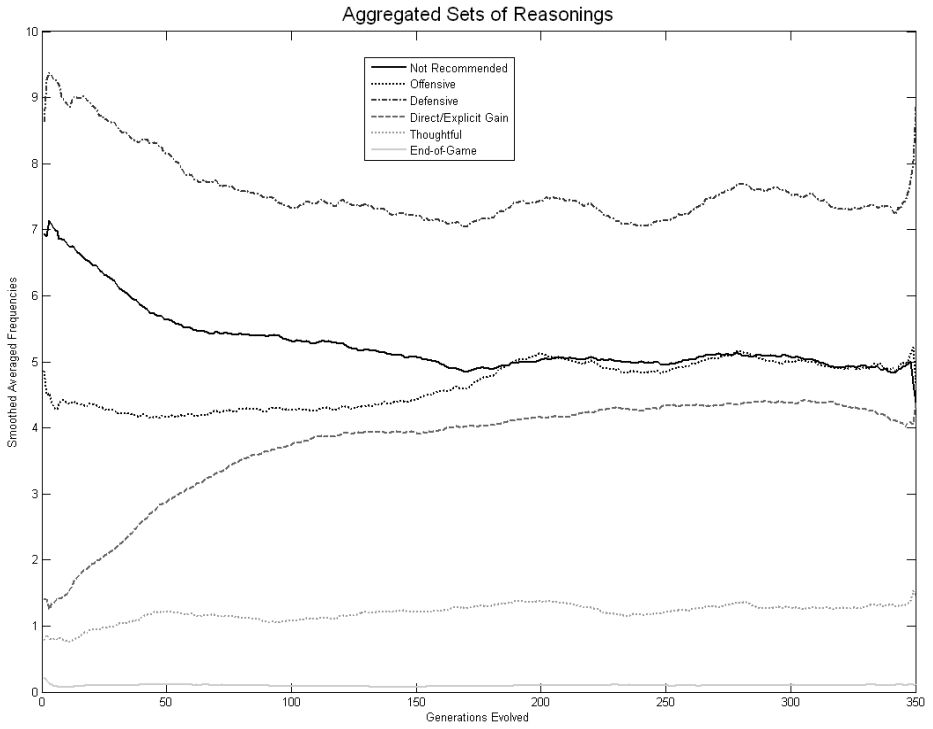


Figure 6.4. Aggregated Sets of Reasoning; The Overall Strategies.

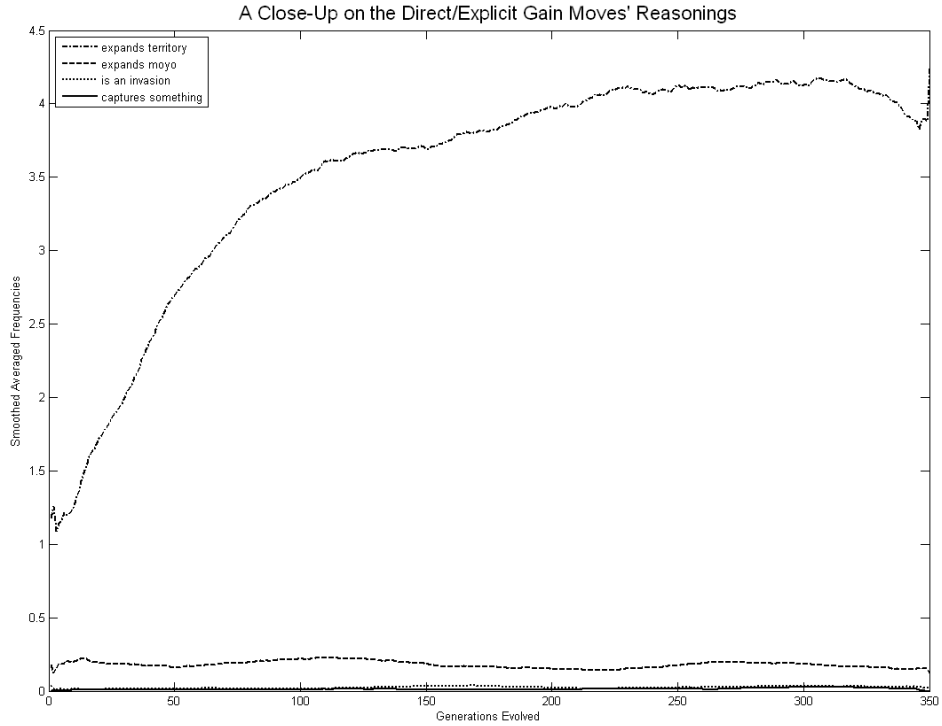


Figure 6.5. A Close-Up on the Direct/Explicit Gain Moves' Constituent Reasoning.

Overall, the reported figures provided a semantic monitoring of the neuro-Go players during evolution by explicating the strategies and reasoning behind the moves played by the network. This demonstrates our initial proposition that the established computational framework can unfold the evolution of a NN player. The next section put the last touches on this demonstration by combining those generated reasoning with a data-mining technique to provide a probability for the evolution process to converge. This probability will offer an alternative indication of possible stagnations.

6.3.2 Dynamics of the Neuro-Evolution

The trend of the evolution process can be – *more or less* – noticed from Figures 6.3, 6.4, and 6.5. The reason why we need to separately judge the dynamics of the neuro-evolution – *rather than depending merely on the results in Figure 6.2* – is that even though the performance criteria is not improving, the networks might still be in the middle of adopting new strategies that are not yet reflected in the performance. In such a case, the strategies that are being adopted would need more evolution to become established and effective.

Therefore, from Figures 6.3, 6.4, and 6.5, a relatively flat curve – *plateau* – can be noticed for the generations beyond *150*. However, we can use the RF trained to classify immature versus mature games to describe the overall neuro-evolution dynamics. Figure 6.6 shows the results for *17500* games, representing *5* games played by the best network in each of the *10* runs, and – *temporally* – for the *350* generations. The probability *1* indicates an extremely untrained – *randomly playing* – network, while *0* indicates a network with game-playing strategies comparable to those of the opponent *(i.e., GNU Go at level 1)*. The evolution trend noticed before is reflected by the probability curve. The curve drops by *0.07* in the first *8000* games – roughly corresponding to the *150* generations, and for the rest of the games, the probability drops by only *0.01*.

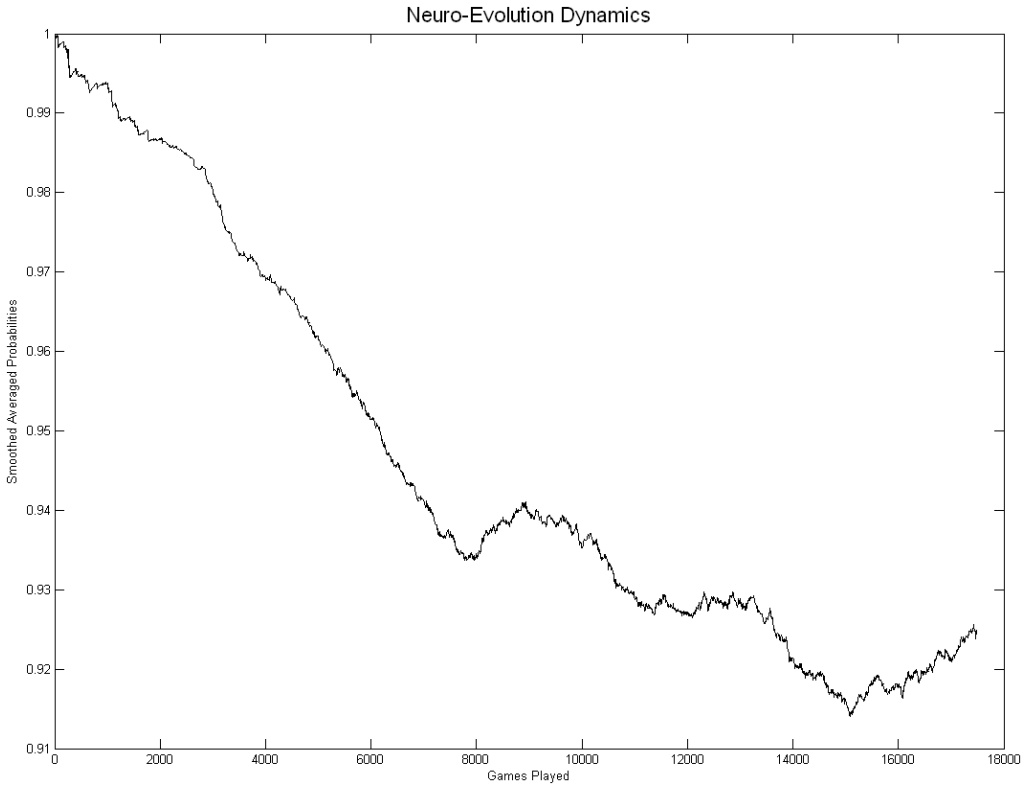


Figure 6.6. The Probability of being an Immature (Untrained) Player.

6.3.3 Hinton Boards

Concurrent with the previous methodology, we used modified Hinton Diagrams for board games, as an independent validation approach for the neuro-evolutionary dynamics. In this section, we present Hinton Diagrams that provide a qualitative display of the evolving networks' weights values. Hinton diagrams traditionally displays the weights of a network in terms of squares with sizes that are relative to the magnitude of each individual weight (Unluturk et al., 2011).

Instead of the traditional Hinton Diagrams, in which the axes represent the connecting nodes, the values were added and organized in away corresponding to a Go board. Those Hinton Boards present the weight values of the connections from the *500* node hidden layer to the *81* node output layer. Figures 6.7, 6.8 and 6.9 display the average of the best network in the *10* runs at generations number *0*, *150*, and *350*, respectively.

The figures show how the weight values that connect to the output layer adapt with evolution. In Figure 6.7, the positive *(light grey)* and negative *(dark grey)* seem to be equally distributed throughout the board. By the *150th* generation (Figure 6.8), besides increasing in magnitude, the positive weights become concentrated in the centre area of the board, while the borderline is usually occupied by the negative weights. A trend that continued to develop – yet without the same initial momentum – until the *350th* generation (Figure 6.9).

The later statement may reflect the fact that by the *150th* generation, the performance enters a relatively flat plateau. The observation that the positive weight values are concentrated towards the inside of the board, clearly reflects a primitive approach to developing and increasing one's own territory by playing towards the centre and of diminishing play at the borders of the board.

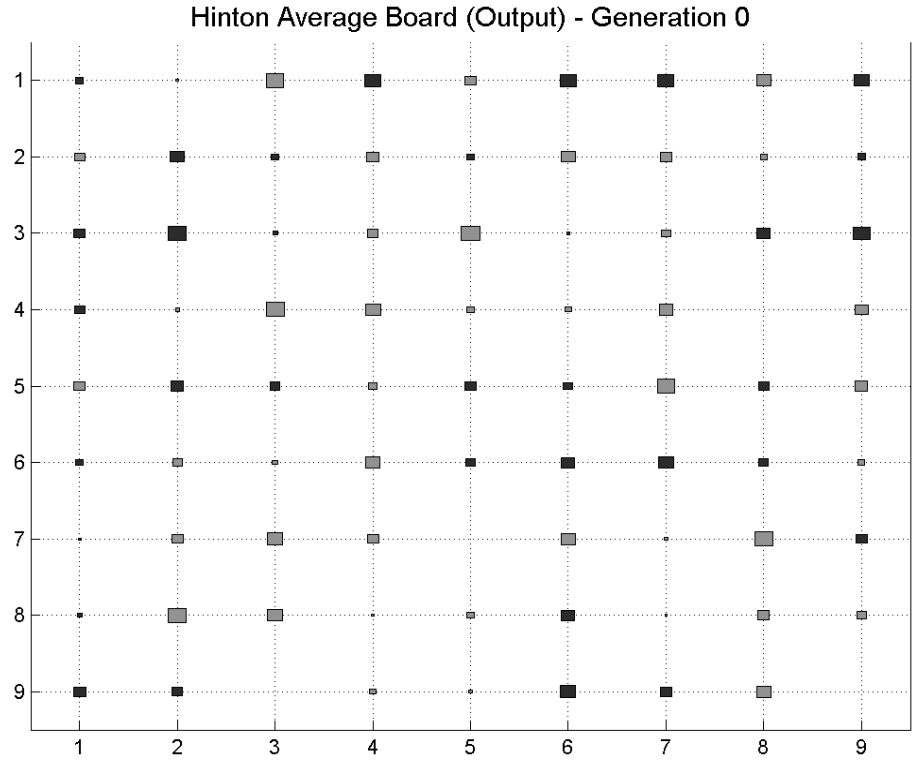


Figure 6.7. Averaged Hinton Board for the output-connections weight-values, generation number *0*.

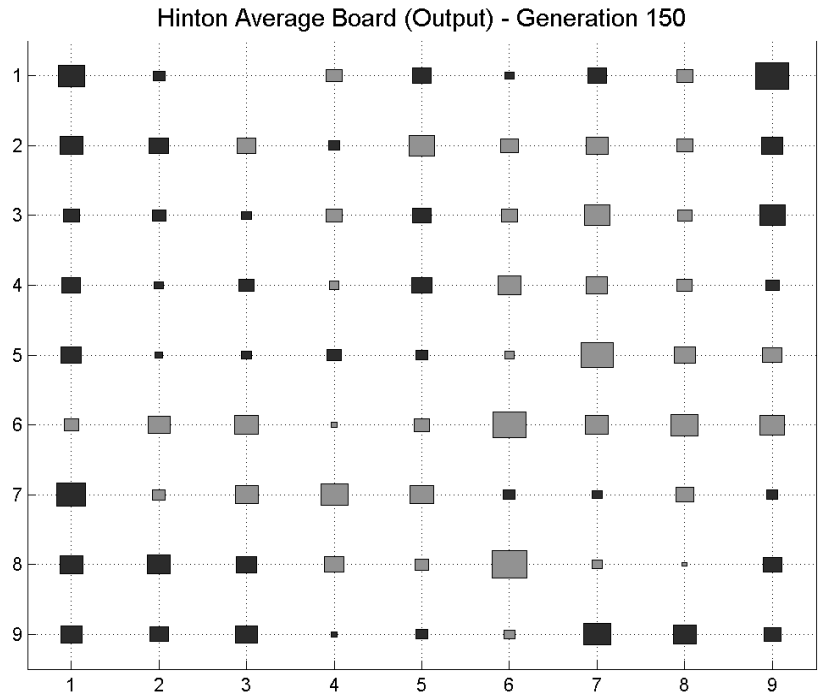


Figure 6.8. Averaged Hinton Board for the output-connections weight-values, generation number *150*.

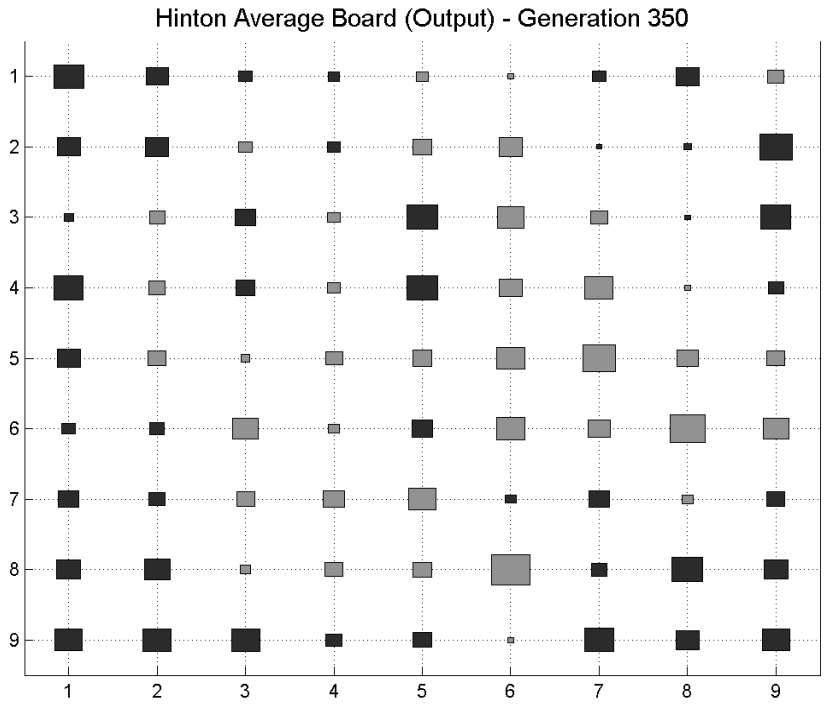


Figure 6.9. Averaged Hinton Board for the output-connections weight-values, generation number *350*.

6.4 The Need to Integrate Semantic Knowledge into the Neuro-Evolution

What is necessary to change a person is to change his awareness of himself.

–––Abraham H. Maslow (1908 – 1970)

Integrating domain and expert knowledge, human preferences, and other forms of priori knowledge into evolutionary algorithms (EAs) has been widely investigated in recent years (Jin, 2004; Bonissone et al., 2006; Kim and Cho, 2007). Bonissone et al. (2006) discussed the implicit (e.g., Data Structures, Encodings, and Constraints) and explicit (e.g., controlling the EAs parameters and Seeding Initial Populations) mechanisms for evolutionary computation, and the benefits of such mechanisms to real-world optimization problems. In light of the No Free Lunch Theorem that states that an algorithm's performance – and thus, its ability to outperform another – is limited by the degree of integrated knowledge that is related to the cost function (Wolpert and MacReady, 1997), and consequently, by the degree that the algorithm is focused on a particular problem (Ho and Pepyne, 2002). Bonissone et al. (2006) concluded that, by integrating domain knowledge, the application of EAs to solve complex real-world problems can be significantly improved.

In this remaining part of this Chapter, we propose a novel methodology for the integration of domain knowledge into EAs for evolving a complex application; that is, in particular for Go players. As discussed before in Sections 2.2.4 and 6.2, neuro-evolution when applied to board games – *specifically Go* – still suffers from considerable disadvantages. Beside the enormous computational cost that limits the sizes of the boards being evolved to 9×9 (with 5×5 as the typical size utilized in the literature), many of the evolved players lack common high-quality playing capabilities. Table 6.2 shows a summarization of the related-work in applying neuro-evolution to the game of Go.

Table 6.2. Summarization of the Related-Work in Applying Neuro-Evolution to the Game of Go.

Reference	Approach	Board Size	Opponent (If Applicable)	Number of Generations	Output
Richards et al. (1998)	- Symbiotic Adaptive Neuro-Evolution (SANE); Evolving two populations, one of neurons and another of blue-prints.	5×5, 7×7, and 9×9	*Wally* (a trivial computer player)	20, 50, and 260 generations respectively	- Able to defeat *Wally*. - Took advantage of Wally's weaknesses. - Displayed some characteristics of common Go playing.
Lubberts and Miikkulainen (2001)	- Competitive Coevolution; coevolving two populations, one of optimal solutions and another of test cases.	5×5	-	40 generations	- Better quality that was not limited by the opponents.
Mayer and Maier (2005)	- Cultural Co-evolution; coevolving NNs while dynamically growing a group of the master players that gathers and saves the knowledge of the population.	5×5	-	Up to 55,000 generations	- Tested against a Random Player, a Naïve Player, and *JaGo*.
Stanley and Miikkulainen (2004)	- Roving Eye; a visual field more limited than the board but which could scan the board at will. - Using NEAT (Neuro-Evolution of Augmenting Topologies), Roving Eyes NNs were evolved on a 5×5 board, then further evolved on a 7×7 board.	7×7	*GNU Go*	500 generations	- Better performance than learning directly on the larger board.

A step towards addressing this problem was accomplished by utilizing a computational approach to capture the nature of the strategies evolved. We propose to evolve NNs with high-quality playing capabilities by integrating domain knowledge about playing strategies into the fitness function. Currently, in the Go literature, fitness is calculated depending entirely on the final outcome of the games played during the evaluation phase, i.e., the games' scores. Recently, using Checkers and Othello board-

games, domain knowledge has been added to EAs (Kim and Cho, 2007). In next section, we propose a multi-objective fitness measure that adds a strategically-aware component to the traditional score.

6.5 Proposed Methodology: Guiding the Neuro-Evolution of Computer Go Players

Our basic neuro-evolution algorithm, and how it can be applied to evolving Go players, was defined in Section 6.2.1. In addition, we need to clearly state the concerns we are trying to address. This section starts by defining the problem, and then describes how a strategic knowledge component can be achieved, followed by how this component can be added to the traditional fitness function. This section ends with the experimental setup.

6.5.1 Problem Description: A Strategically Aware Fitness Measurement

From the previous literature review, one can conclude that the neuro-evolution of Go players currently has three 'conflicting' objectives/Concerns; namely, decreasing the computational cost, evolving better strategies and avoiding stagnation. Our goal is to investigate the significance of a strategically-aware fitness function in achieving those objectives.

To accomplish this goal, SANE was used to evolve 9×9 neuro-Go players using both the traditional exclusively score-dependent fitness function, and the proposed strategically-aware fitness function. It is noteworthy that evolving a standard *19×19* capable worthy player is estimated to cost tens of thousands of generations with a probable CPU time of up to a year (Richards et al., 1998). The networks will be evolved against GNU Go as an opponent. To compensate for the additional computational cost of estimating the strategies in the Go games, *50* blueprints are evolved instead of the *200* suggested by (Richards et al., 1998). Due to the nature of the problem in which a network is evaluated by playing a game, and in spite of using elitism, the fitness values

– *across the generations* – are expected to fluctuate due to the varying performance – *of a single network* – in different games.

6.5.2 The Domain (Strategic) Knowledge Component

The strategic competent corresponds to the output of a Random Forest (RF) that is trained to classify a game of Go as either immature/untrained or mature/fully-trained, via a *'Trained Probability'* (TP). For this, an untrained network shows no Go playing strategies, while a fully-trained network shows considerable strategical capabilities. The value of the TP is *0* if the investigated network is untrained, and *1* if the network is fully trained. The input features to the RF are the *'Reasons Frequencies'*; the frequency of each reason per game. The *Reasons Frequencies* (Eq. 5.1) are calculated based only on the moves played by the Neuro-Go player being evaluated; that is, using only half of the game. The RF is trained using two databases (Figure 6.10) of features, each containing *150* games. The untrained games were selected from *150* games played by NNs evolved in the very first *3* generations, therefore practically showing no game-playing capabilities. On the other hand, the fully-trained games were selected from games played by advanced players on the Computer Go Server (CGOS, 2011).

After generating the Strategically-Aware component – i.e., the TP – the traditional score-based fitness function can be modified by simply adding the generated probably to the game score (Figure 6.10). The effect of the added term can be tuned by the coefficient α. The proposed fitness measure *f* for a *Network* is calculated as:

$$f_{Network} = \sum_{i=1}^{N_{Games}} \frac{\alpha TP_i + (1-\alpha)\, Score_i}{N_{Games}} \qquad \text{(Eq. 6.1)}$$

where N_{Games} is the number of games played by *Network* in the evaluation phase, $Score_i$ is a value – in the range from *0* to *1* – representing the *Network*'s score in game *i*, while TP_i is the Trained Probability generated by the RF for the game *i*.

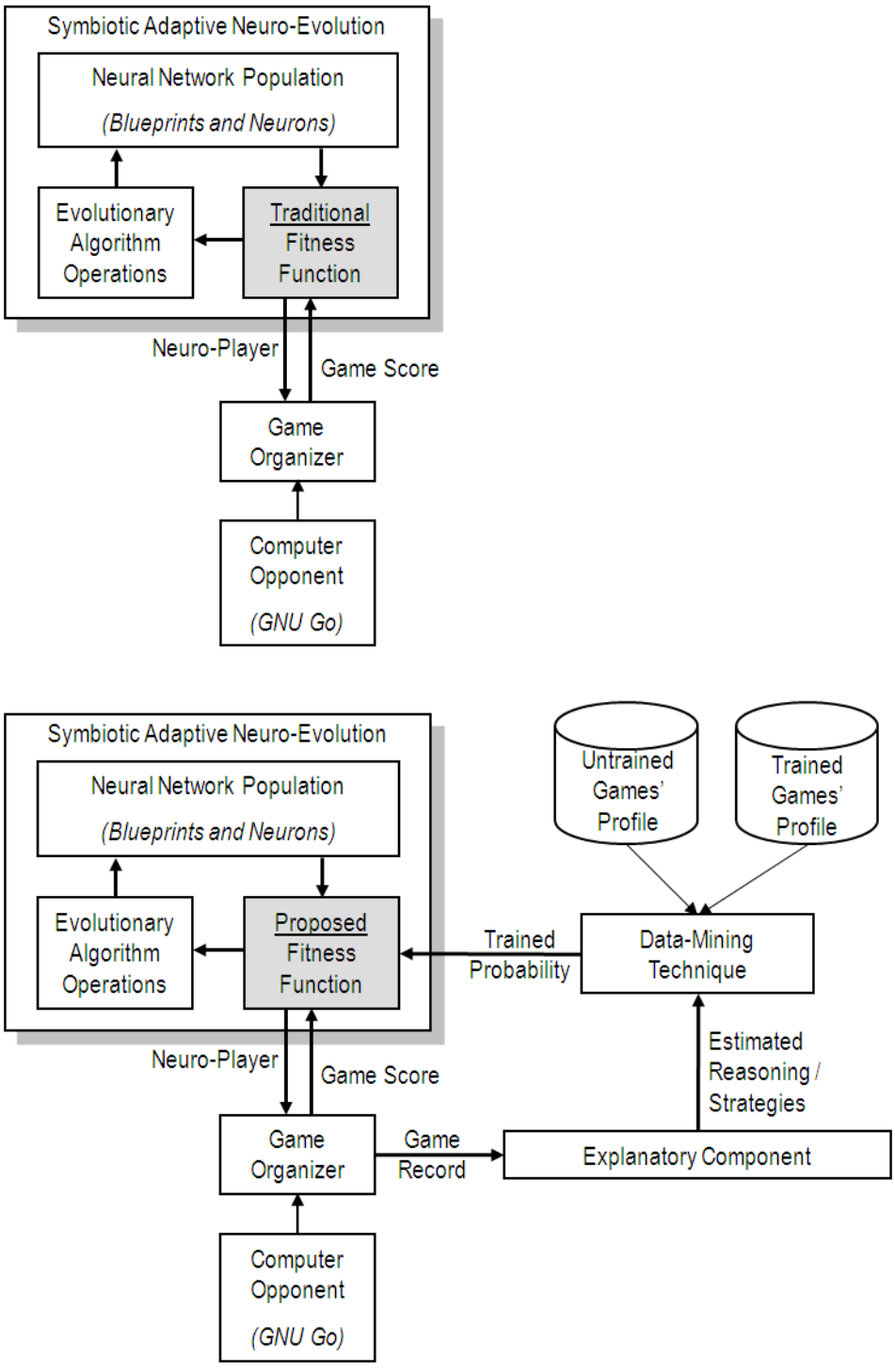


Figure 6.10. The Traditional Score-Based Fitness Function Versus the Proposed Strategically-Aware Multi-Objective Function.

6.5.3 Experimental Setup

To investigate the effect of the added TP, the coefficient α was varied, using three values; *1.0*, *0.5*, and *0.0*. The first α value represent the case where the networks' evaluation is based entirely on the Trained Probability, while α set to *0.0* represents the traditional score-based fitness function.

In General, the parameters in the experiments are based on those found effective in (Richards et al., 1998), except – as mentioned before – for the number of blueprints which was reduced from *200* to *50*. A single run consists of *50* generations, and *10* different runs were evolved. With *5* games played per each fitness evaluation, the total number of games *"being evolved for"* is *2500*. As in Section 6.2.2, instead of GNU Go's default level of *10*, its level was set to *1* throughout the conducted experiments, that is a difference of about *4* stones.

The Go games were scored using Chinese rules. The networks were evolved to play both Black and White, thus frequently making the first move. To avoid a tie, a *komi* value was set to *0.5*. No handicap stones were given to the neuro-players being evolved. The upper bound on the number of moves per game – required to discontinue any irrationally extended sequences of moves – was set to *81*. In the conducted experiments, a RF is an ensemble of *1000* classification decision trees, and the minimum number of observations per tree leaf is *1*.

This experimental setup cost was up to a maximum of ≈10 days for a single run per an α value using a Sun Constellation Cluster. Details about the hardware specifications may be found at (NCI, 2011).

6.6 Results and Discussion

In order to investigate the effect of the proposed fitness function, three different types of analysis – to the neuro-evolution process – are shown and discussed. We start of by showing and discussing the convergence among the varying α values. Then a Playoff between the best players – at each run and generation – is held; from which the

dynamics of the strategies evolved are monitored. Finally, the results of the playoffs – *in terms of the final score* – are presented, including four representative games which are plotted.

6.6.1 Convergence

Figure 6.11 shows the convergence of the *50* blueprints and *2000* neurons evolved using SANE for *2500* games (*50* generations). For each α value, the average fitness – of *10* different runs – for the 1) best network, and the 2) entire population are plotted.

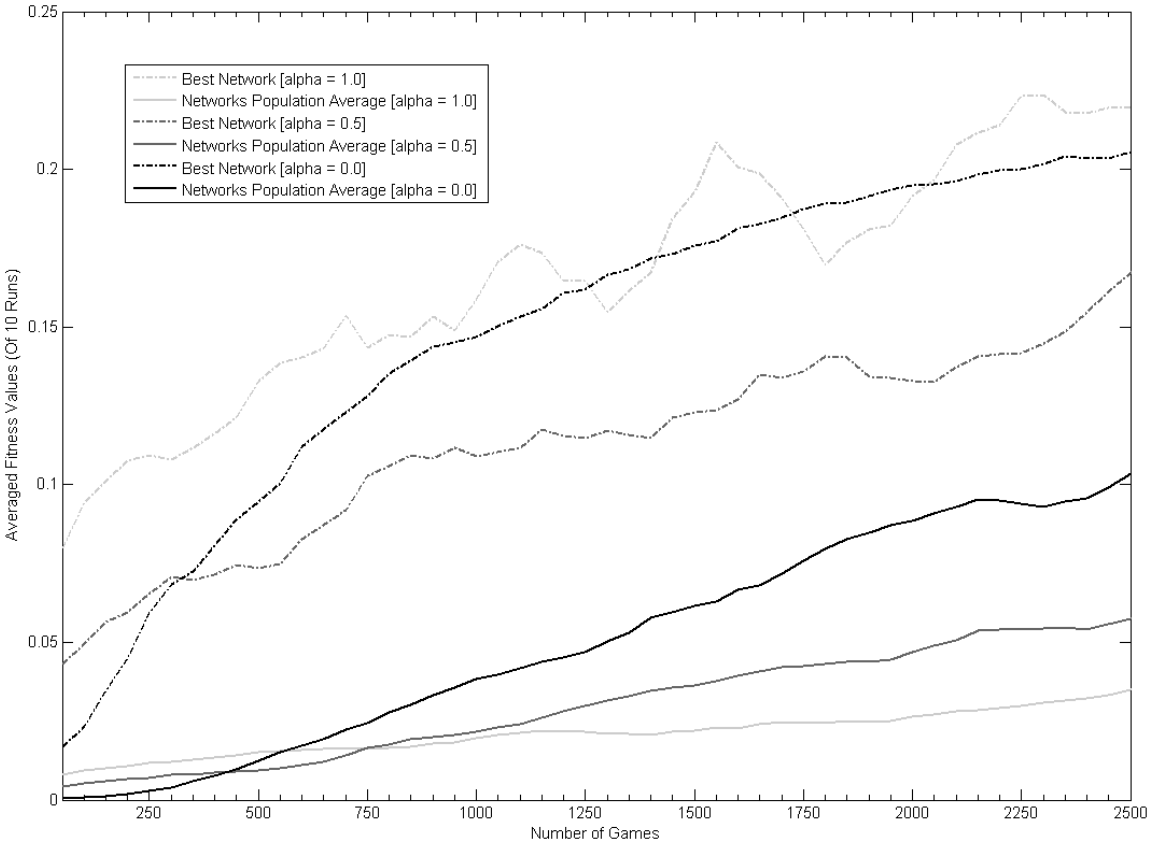


Figure 6.11. The Convergence of the Fitness Values among the *2500* Games, Averaged for the *10* Runs.

The convergence of the fitness values of the best networks for an α value of *0.0* seems to enter a relative plateau, starting from around generation number 45 (around

2200 games), as opposed to both α values of 1.0 and 0.5. On the contrary, the convergence of the entire populations seems to continue evolving. Notably, the 'relative' difference between the best network and the population in terms of fitness values is largest for the α value of *1.0*. A possible explanation is that while depending more on the TP component rather than the score, the evolving networks increasingly fluctuate between the generations.

However, the results for the convergence of the fitness values of the best networks show an advantage for the TP component, when compared to the traditional score-based function, in reducing stagnation. The proposed component evidently, within the investigated generations, better forces the continuous evolution of the neuro-players by guiding the development of enhanced Go-playing strategies. The quality of those Go-playing capabilities developed will be further investigated in the next two subsections.

6.6.2 Playing Capabilities: Analysing the Strategies of a Go Playoff

The first step to investigate the playing capabilities evolved is by holding a playoff between illustrative players of the varying α values. A simple and straightforward criterion is used to select representative players for each of the varying α values; the networks achieving the overall best – *for instance* – 'game score' in each of the *10* runs and in each of the *50* generations. Between each two players, *2* matches were played with both players alternating being *Black*; that is, alternate starting the game. The komi value was set to *0.5* with no handicap stones, and the games were scored using Chinese rules. Figure 6.12 illustrates the details of the accomplished playoff.

Figure 6.13 shows the overall number of reasons behind the moves played during this playoff, for the different fitness function, and as the networks evolved temporally. Figures 6.14 to 6.18 show the reasons as grouped into strategic categories (Table 5.1) for the corresponding players.

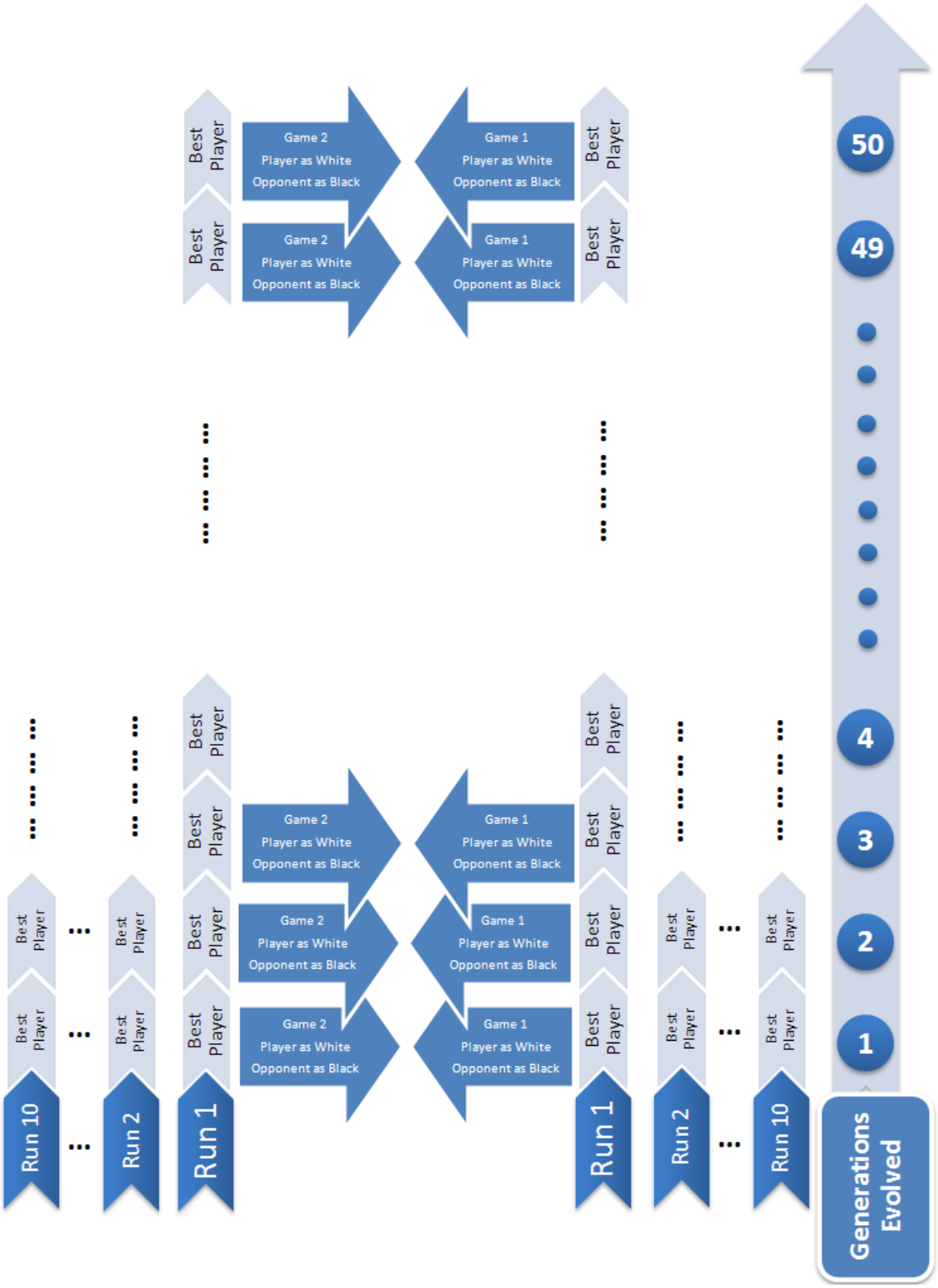


Figure 6.12. An illustrative diagram of the Playoff.

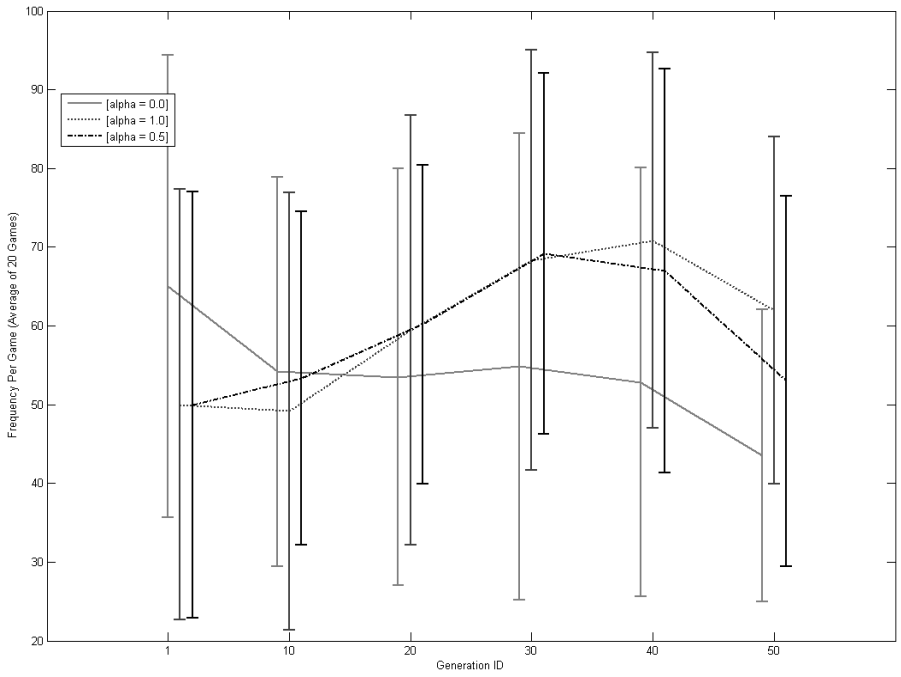


Figure 6.13. The Frequency of Reasoning Per Game – *for All Reasons* – among the *2500* Games, Averaged for the *10* runs, and *2* matches per run.

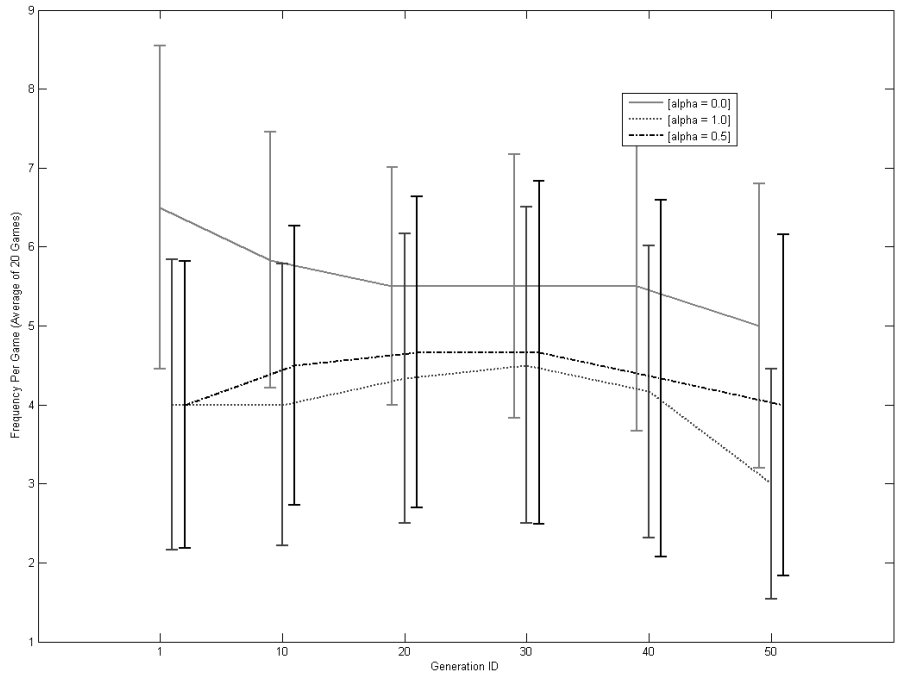


Figure 6.14. The Frequency of Reasoning Per Game – *for Not-Recommended moves* – among the *2500* Games, Averaged for the *10* runs, and *2* matches per run.

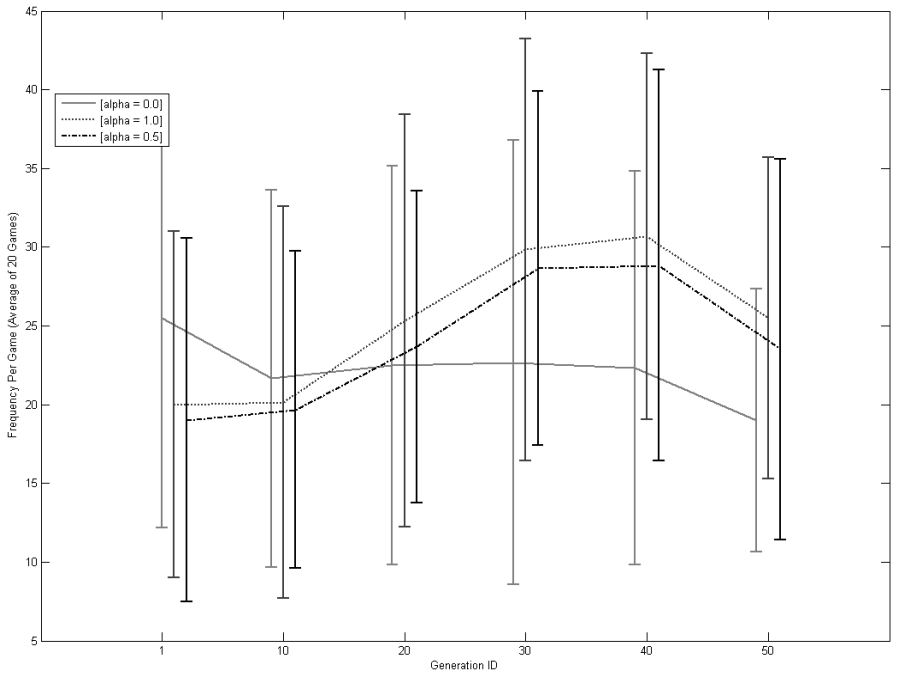


Figure 6.15. The Frequency of Reasoning Per Game – *for Defensive moves* – among the *2500* Games, Averaged for the *10* runs, and *2* matches per run.

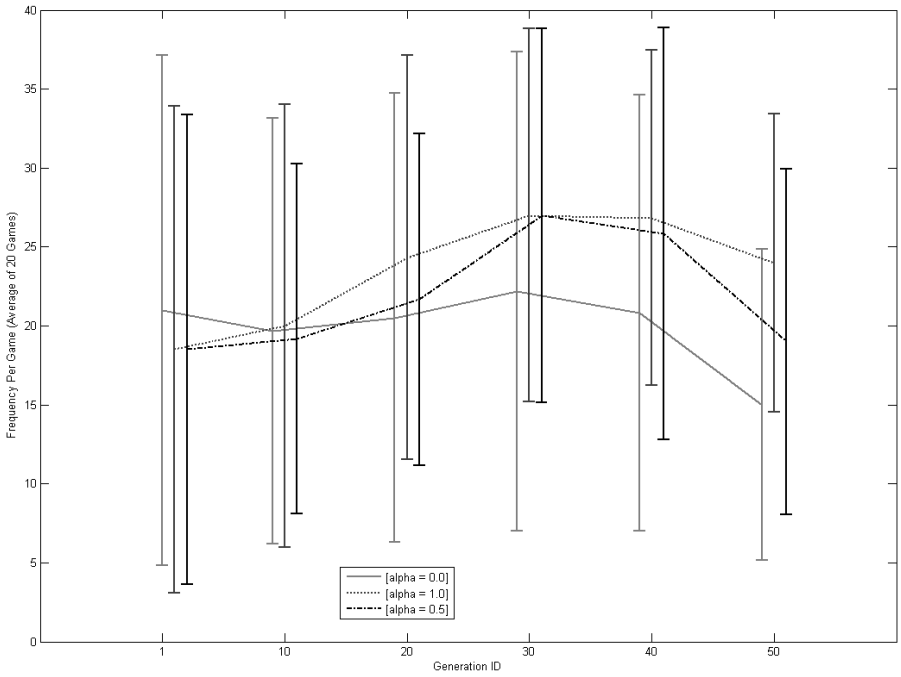


Figure 6.16. The Frequency of Reasoning Per Game – *for Offensive moves* – among the *2500* Games, Averaged for the *10* runs, and *2* matches per run.

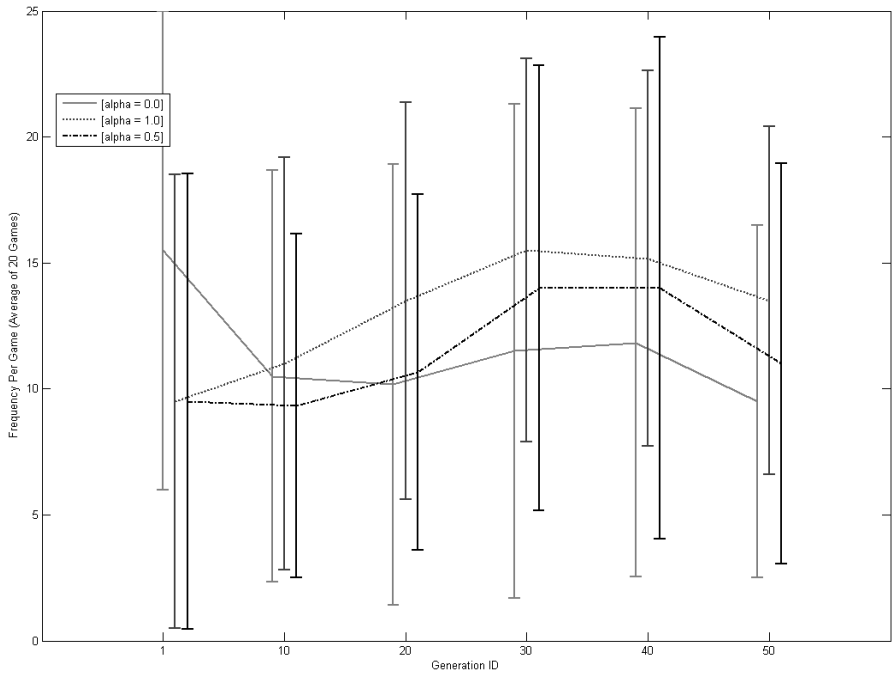


Figure 6.17. The Frequency of Reasoning Per Game – *for Thoughtful moves* – among the *2500* Games, Averaged for the *10* runs, and *2* matches per run.

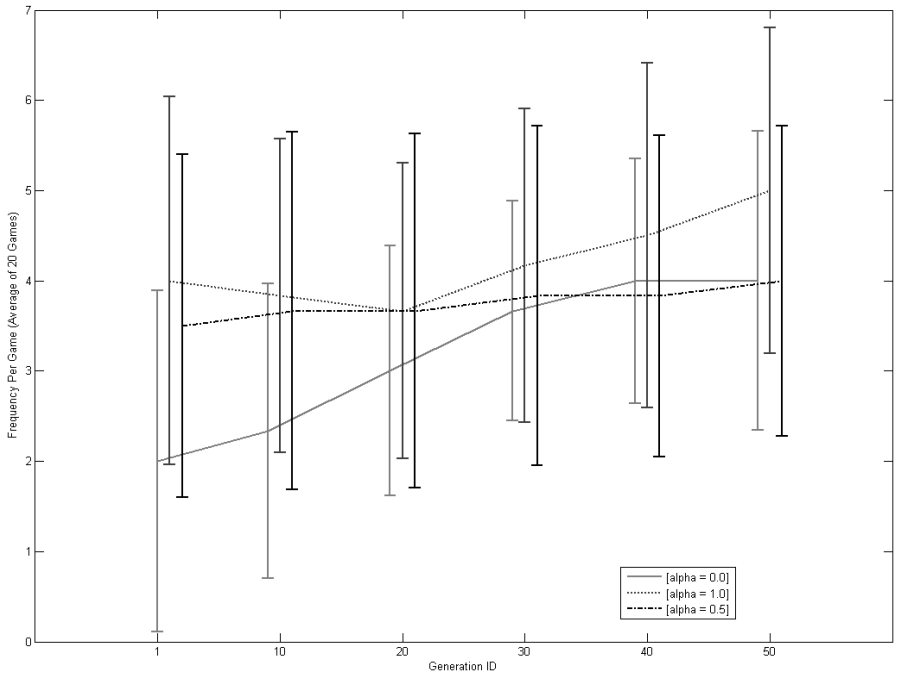


Figure 6.18. The Frequency of Reasoning Per Game – *for Explicit-Gains* – among the *2500* Games, Averaged for the *10* runs, and *2* matches per run.

Figure 6.13 clearly shows that the overall number of reasons increased for the players evolved using the proposed fitness component, whether solely (i.e., for $\alpha = 1.0$) or partially (i.e., for $\alpha = 0.5$). As for the players evolved using the traditional scoring only (i.e., for $\alpha = 0.0$), the overall reasoning did not change temporally, and by the end of the *50* generations evolved, it stood at $\approx 70\%$ of the reasoning behind the moves played by its opponents. This is reflected in all of the subsequent figures, except for Figure 6.14 which represents the *Not-Recommended* Moves. As shown in that figure, the not-recommended reasoning reported for the played moves were lower when the strategically-aware component was included. As for the rest of the strategic categories – that is, Offensive, Defensive, Thoughtful, and Explicit Gains – the reported reasoning was higher when using the proposed component in comparison to the traditional function. It is worth mentioning though that in Figure 6.18, the difference between the varying α values is minimal, that is obviously due to the pressure exerted by the traditional fitness function on moves that tangibly/directly capture stones and/or invades and secures an opponent's territory. The category 'End-of-Game' (Table 5.1) – *representing small moves that occur late at the game* – was omitted because virtually no substantial frequencies-of-reasons within this category were reported. This is a clear indication that those strategies are yet to be developed in later generations by the evolving neuro-players.

Those reported results demonstrate that the strategically-aware fitness function provided better assessment for the neuro-players, and consequently, led to the evolution of players with improved Go-playing capabilities. This improvement, when compared to the traditional score-based fitness, is evident in terms of the overall number of reasons behind the actions taken by the players within the game, and also in terms of the type strategies being deployed. Overall, this demonstrates a positive value of enhancing the competency awareness of the developed – i.e., evolved – playing agents, which is a direct consequence of the proposed automated competency assessment approach. These conclusions will be further demonstrated in the upcoming sub-section, in which the scores of the playoff will be presented.

6.6.3 Playing Capabilities: Analysing the Scores of a Go Playoff

Figures 6.19 to 6.21 show the scores of the conducted playoff. Notably, the improved reasoning evolved within the players – of both $\alpha = 1.0$ and 0.5 – is reflected temporally with relatively higher scores when playing against traditionally-evolved players (i.e., with an $\alpha = 0.0$). Figures 6.22 to 6.27 show the details of four representative games from the playoff. For the middle run (*Run # 5*), Figures 6.22 and 6.23 show – respectively – details of the largest win *(+20.5)* and lose *(-11.5)* for an $\alpha = 1.0$ player against a traditional $\alpha = 0.0$ player. Followed by Figures 6.24 and 6.25 presenting, from the same run, the results for the final *2* games *(at the very last generation evolved)* played by an $\alpha = 1.0$ player against a traditional $\alpha = 0.0$ player; as white *(winning by 10.5)* and as black *(losing by 0.5)*. Finally, Figures 6.26 and 6.27 illustrate selected interesting moves within the later *2* games.

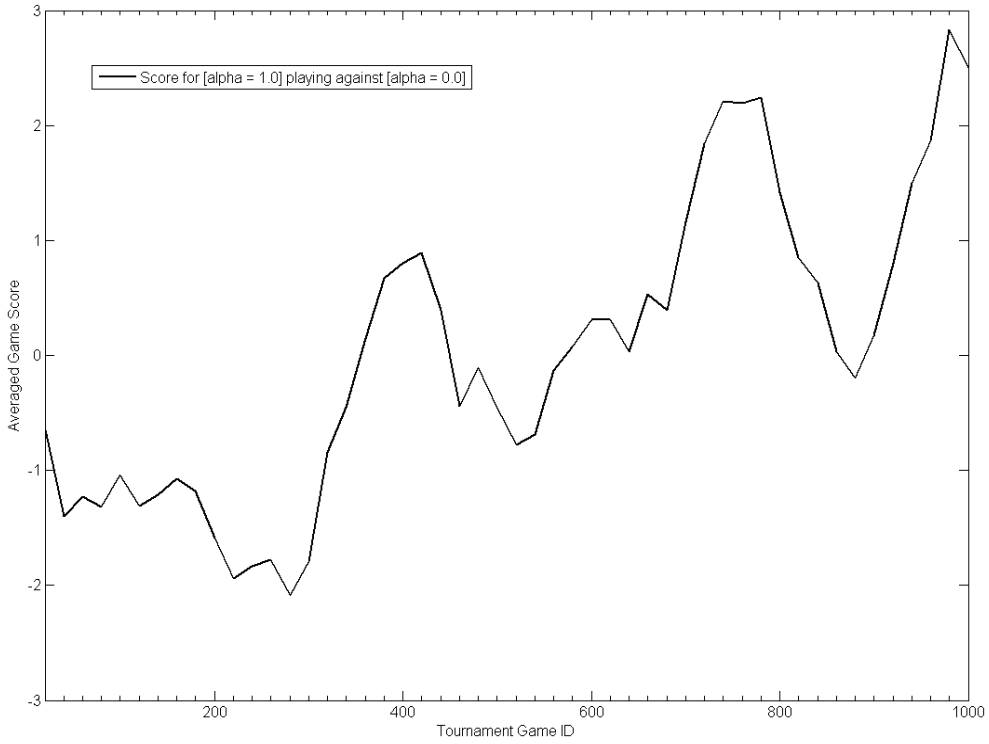


Figure 6.19. The Scores for $\alpha = 1.0$ *(networks evolved using the proposed fitness function)* playing against $\alpha = 0.0$ *(networks evolved using the traditional fitness function)*.

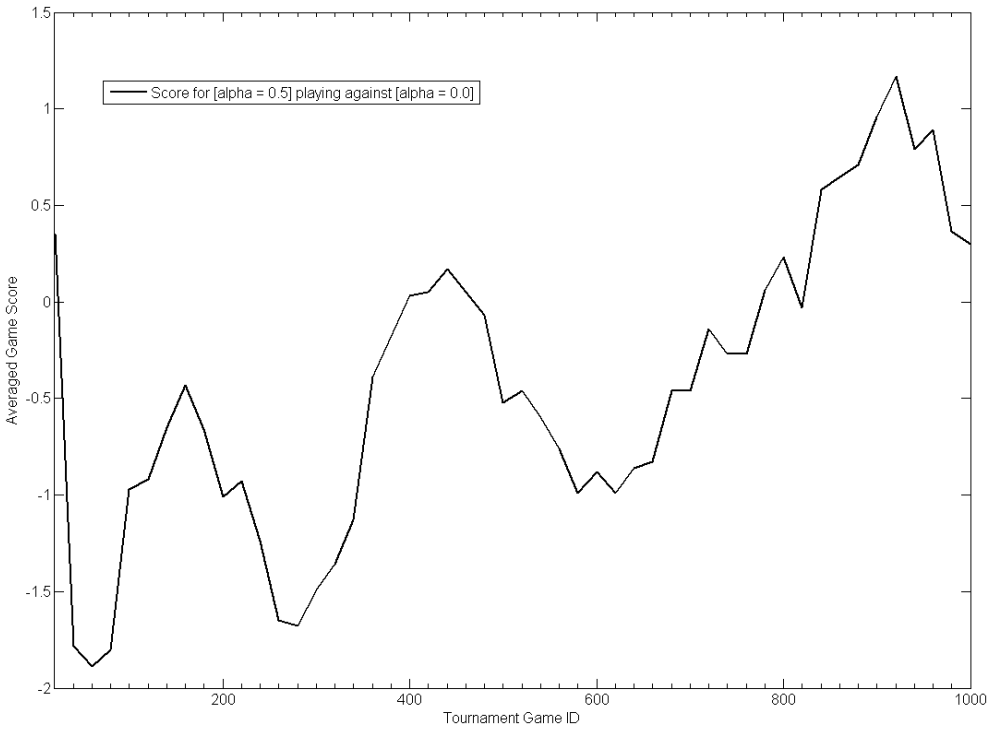


Figure 6.20. The Scores for $\alpha = 0.5$ playing against $\alpha = 0.0$.

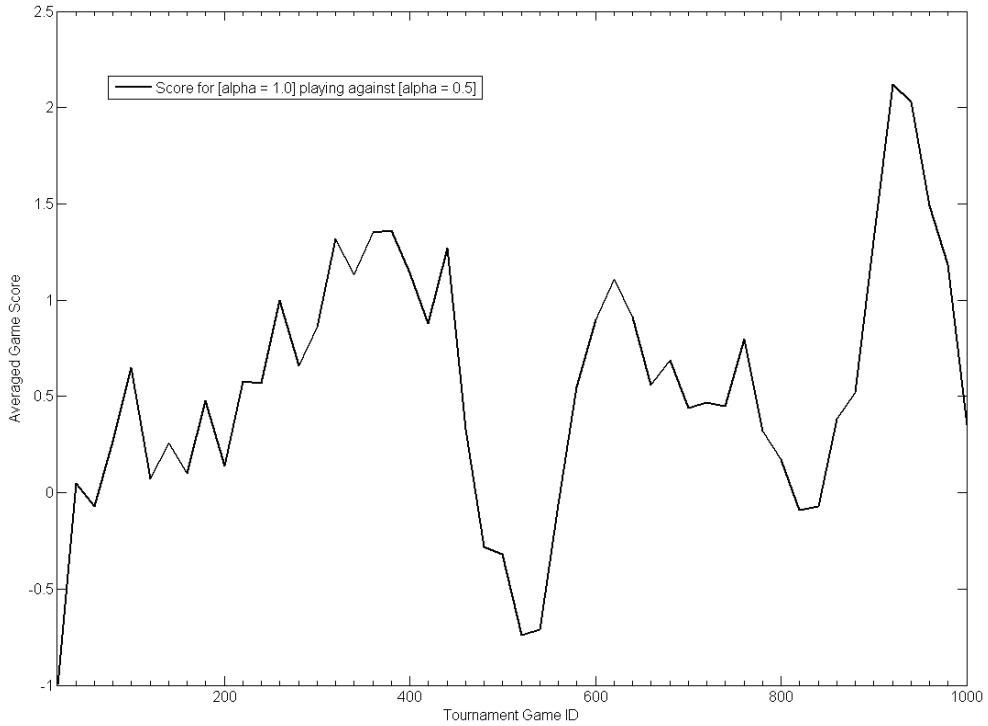


Figure 6.21. The Scores for $\alpha = 1.0$ playing against $\alpha = 0.5$.

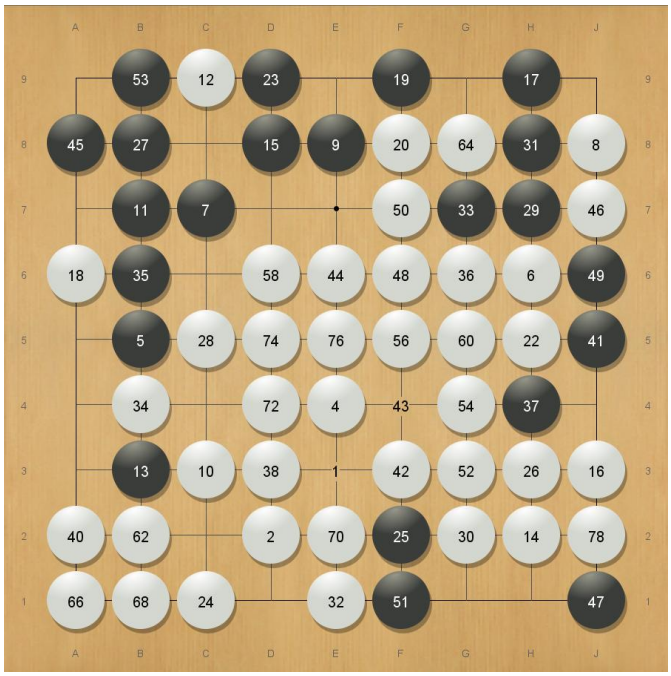


Figure 6.22. The largest win *(20.5)* of a player *(playing as White)* evolved *(in Generations # 35, Run # 5)* using $\alpha = 1.0$ against $\alpha = 0.0$.

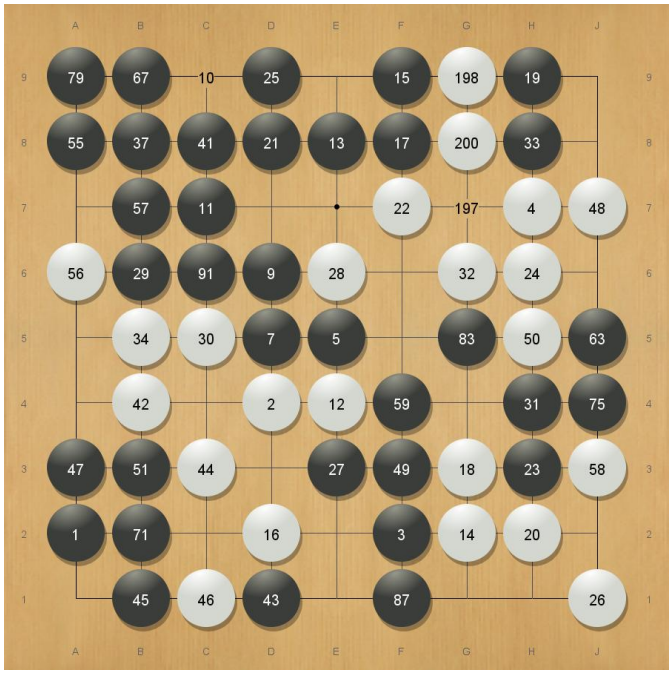


Figure 6.23. The largest win *(11.5)* of a player *(playing as Black)* evolved *(in Generations # 24, Run # 5)* using $\alpha = 0.0$ against $\alpha = 1.0$.

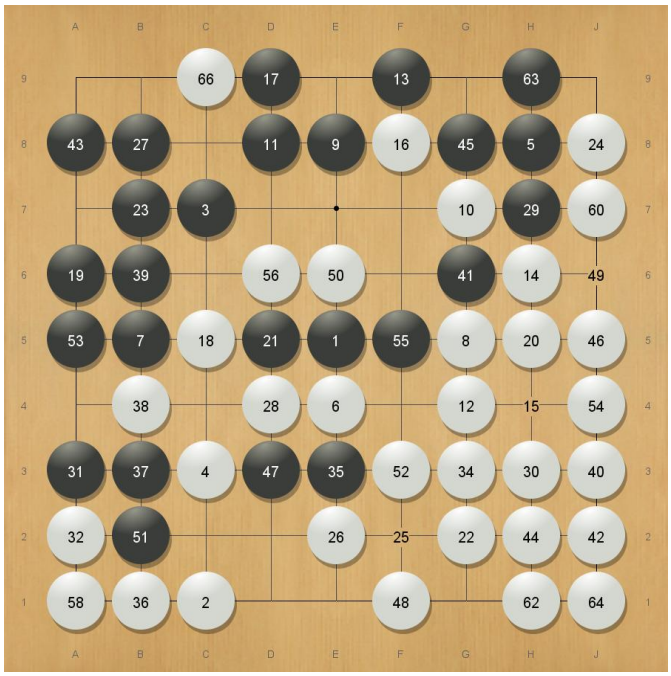


Figure 6.24. At the Final Evolved Generation *(# 50)*; a player *(playing as White)* evolved *(in the Run # 5)* using $\alpha = 1.0$ against $\alpha = 0.0$ wins by *(10.5)*.

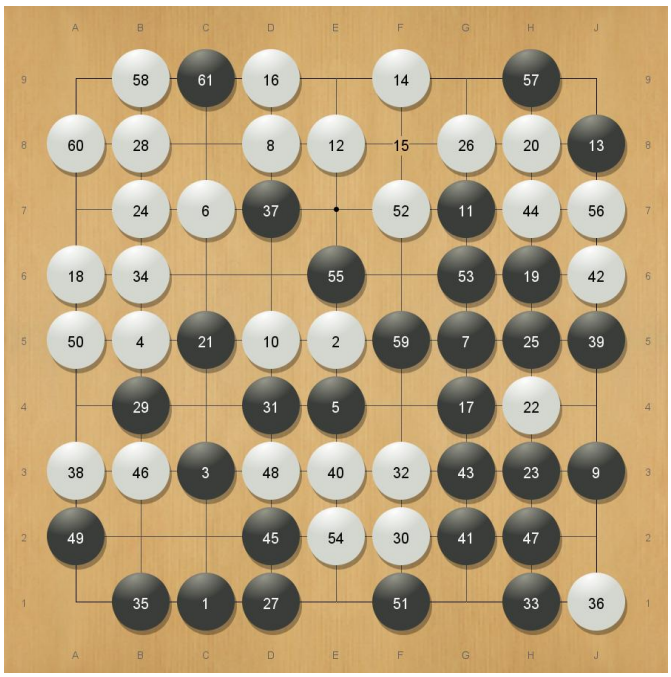


Figure 6.25. At the Final Evolved Generation *(# 50)*; a player *(playing as Black)* evolved *(in the Run # 5)* using $\alpha = 1.0$ against $\alpha = 0.0$ loses by the komi value *(0.5)*.

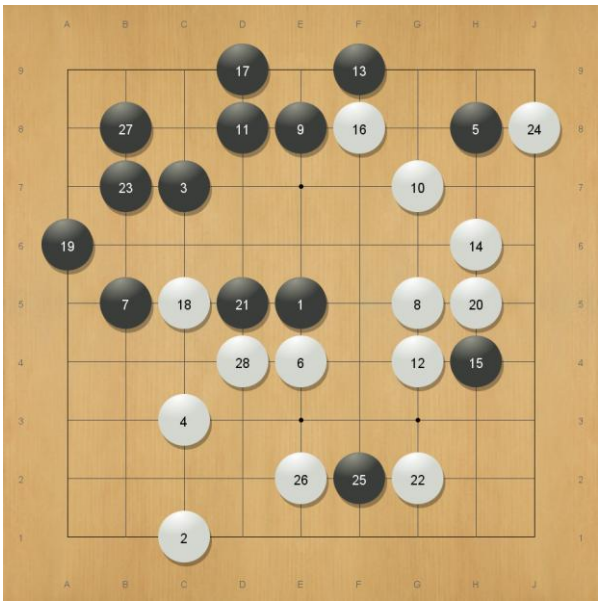


Figure 6.26. The White move at D4 – *from Figure 7.14, numbered 28* – defends *C5*, connects *E4* and *C3*, also connects *C5* with both *E4* and *C3*, and thus it is also considered an expansion to White's territory.

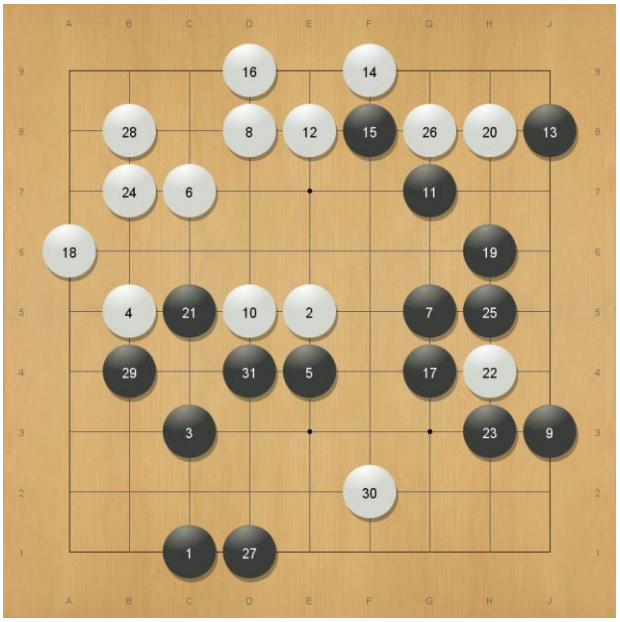


Figure 6.27. The Black move at D4 – *from Figure 7.15, numbered 31* – Connects the following stones to each other *[B4, C3, C5, E4]*, strategically defends *C5*, and is an expansion of Black's territory.

6.7 Summary

In this chapter, in order to address the third sub-question (Section 1.3), we presented a novel approach to address the disadvantages of neuro-evolution in mind-games. Namely, the limited quality of the solutions that were evolved and stagnation. In comparison to the traditional rule-extraction techniques, instead the proposed framework was used as an explanatory model to capture the types of game-playing strategies evolved. Augmented with a data-mining technique *(i.e., Random Forest)*, the proposed approach successfully captured the overall evolutionary dynamics.

From the reported results, it was observed that the number of reasonless moves decreased as the neuro-players evolved. Thus, the types of strategies developing were observed, showing a decrease in the Not Recommended reasoning, and an increase in the Direct/Explicit Gain strategies in which a more detailed close-up demonstrated a quick development of an "expand territory" strategy. Generally, it was also observed that the remaining types of strategies did not vary significantly. Independently, the trained random forest was able to provide a probability curve that reflected the overall neuro-evolutionary dynamics. The results were independently validated using Hinton boards. Those results demonstrated our proposition that the proposed framework is capable of semantically unfolding the dynamics of evolving neuro-Go players.

Based on the resulting observations, in order to address the fourth sub-question (Section 1.3), we introduced a strategically-aware multi-objective fitness function that integrates strategic domain knowledge in to the traditional score-based fitness. Using varying extents of this integration, we have presented experiments evolving Neuro-Go players using Symbiotic Adaptive Neuro-Evolution for 9×9 boards. The strategically-aware component was measured using the previously proposed competency measurement approach *(i.e., Estimated Reasoning coupled by a Random Forest trained to differentiate between strategically trained and untrained Neuro-Go players)*. When compared to the traditional fitness function, the proposed function proved effective in evolving Neuro-Go players with varying strategies that can 1) play more rational and practical moves and, 2) attain better performance against opponents evolved using the

traditional fitness evaluation, and most importantly 3) developed more mature game-playing capabilities, instead of merely exploiting the weaknesses available in an opponent.

The resulting observations have significant implications on future work in applying neuro-evolution to the game of Go, and mind-games in general. The semantically-monitoring approach can – *for instance* – be utilized in 'temporally' selecting the appropriate opponents to influence the strategies being evolved, or by adjusting the co-evolution parameters whenever stagnation is noticed. Consequently, this will assist in the efficient usage of the enormous computational cost usually required for such experiments. The findings are also seen as an advancement towards the creation a better integration of domain knowledge into EAs in general, and also as an improved Neuro-Evolution process for evolving Go players in particular. More importantly, the findings demonstrate the hypothesized value of an objective, quantitative, detailed and fully-automated competency assessment. Namely, attaining and enhancing the competency awareness of a decision-making agent – *within the context of Go*, and consequently, improving the training and development of those decision-making skills within that agent.

[This Page is Intentionally Left Blank]

Chapter 7: Conclusion

> *To tell a truly compelling story, a machine would need to understand the "inner lives" of its characters. And to do that, it would need not only to think mechanically in the sense of swift calculation, but experientially in the sense of having awareness.*
> *---Selmer Bringsjord, Chess Is Too Easy (1998)*

7.1 Summary of Results

This thesis presented a systematic study of the role and value of an automated – *computational* – competency assessment of a decision-making agent in a complex scenario. A computational framework for competency evaluation and awareness was proposed to both investigate the development of expertise in agents and to also assess that development. For this investigation, the game of Go was selected, and the study was carried out using two board-sizes of the game (the standard *19×19*, in addition to *9×9*), and two sorts of agents (Human Go players, in addition to Neuro-Evolved Go engines). During the study procedures, the actions/decisions within the Go space were divided into subsets that represent individual players and their corresponding level of expertise. The Go space was built using hundreds of publicly-available online games in addition to thousands of games generated by – accomplished – evolution-based processes and game playoffs among selected agents. To further ascertain the proposed methodology, the conventional approach of neuro-evolution – *as applied to Go* – was adapted using the computational model, and was compared with the previously-existing version of the approach.

Overall, the main findings of the research introduced in this thesis can be summarized as follows:

1. The concepts of knowledge-acquisition and training bottlenecks were shown to be suitable for characterizing the challenges to computer Go. This representation led to the proposition of an automated approach to competency evaluation – in terms of a computational framework – for further advancements in computer Go.

2. The computational framework offers an effective approach to assess competencies and to provide an enhanced role in the overall awareness and management of complex scenarios. The framework effectively addressed the limitations of the current methodologies for competency assessment.

3. An effective monitoring of the expertise-level in Go can be obtained for human agents on the standard board-size using the proposed framework. The automated assessment of the selected competencies provided an alternative understanding of the nature of expertise in the Go domain. As a result, and augmented with techniques of machine and ensemble learning, the development of such an expertise was monitored across a variety of selected cases. The monitoring was effective for both, the overall level of competency, and for specific constituent skills. A comparison was made that showed the differences between considering the entire interaction between two opponents within a game, as opposed to analysing the actions of each agent individually. In addition, another comparison was made that showed the improved performance of the proposed architecture for the competency-level classifier, as opposed to an alternative architecture that overlooks the concept of an ensemble-learning at its – *i.e., the classifier's* – first tier.

4. The proposed framework also showed a good ability when applied to computer agents and smaller board-sizes to effectively monitor their acquired expertise. This was accomplished by investigating neuro-players evolved to play on 9×9 boards, and thus attempting to address the difficulties of applying neuro-evolution in computer Go. The proposed

framework combined with a data-mining technique (i.e., Random Forests) has been shown to successfully capture the overall evolutionary dynamics. In addition to capturing the nature of the evolved game-playing strategies in more detail; it was contrasted with traditional approaches to representing the activities of neural networks, such as rule-extraction techniques.

5. Based on the results of the proposed framework, guidance information can be provided to the neuro-evolution in terms of a strategically-aware multi-objective fitness function. The proposed fitness function alters our awareness of the evolved neuro-players by integrating domain knowledge to the traditional score-based fitness value. A comparison was made between the traditional and the proposed fitness functions, showing the effectiveness of the later in evolving improved neuro-Go players; in terms of enhanced strategic reasoning that translated to a better performance against opponents.

6. Through this study, the experimental results showed the significance of an automated competency evaluation, as well as situational reasoning, and – *to a lower extent* – connectivity-patterns, as competences capable of capturing the development of expertise. The study also showed the viability and practicability of extending the roles of available computer players – *engines* – to serve as models of situational reasoning.

7.2 Future Directions

There are several possibilities to extend the research presented in this thesis:

- **Investigating different approaches to address Situational Reasoning in the Go domain:** As reviewed in Section 5.1.1, situational reasoning within the Go domain – *and the generic literature* – has been addressed by varying approaches. A future direction may include investigating those approaches *(for instance, the automatic generation of Go rules (Cazenave, 2002))*, and modifying some of the existing approaches

(including the potential of modifying the open-source publicly-available GNU Go engine which has been employed in this study). In addition to investigating these approaches, other modifications might include the provision of additional types of reasoning, covering more specific situations within the domain, and allowing varying degrees of generality and/or detail to the generated reasoning.

- **Addressing the Reduction of the Computational Effort required for Reasoning is a critical objective:** For a more effective application of the proposed framework, especially when coupled with computationally-expensive methodologies such as neuro-evolution, an imperative and pressing concern is the minimization of the computational effort required by the Assessment Component (AC_R). Considering and tackling this issue would allow the proposed framework to be analysed more comprehensively, and also be applied more efficiently. One way might be by using - for instance - notions such as Simulated Annealing to link the results from analysing *9×9* boards to the standard *19×19* boards.

- **Considering Larger Sizes for the Motifs being investigated:** As mentioned before, network motifs, according to definition, are small sub-graphs that are over or under represented within a considered network when compared to randomized networks of the same characteristics. However, patterns of sizes bigger than the smaller ones frequently used in the motifs' literature (usually from 3 to 5 nodes) can be considered. In a preliminary step, *51* and *173* motifs of sizes *6* and *7* respectively were found in the games of *testDS*, out of a total of *112* and *853* possible undirected sub-graphs for those sizes.

- **Considering Directed-Motifs instead of the Undirected Motifs:** Within this study, only undirected motifs were considered. However, a significantly larger number of possible directed motifs exist and can

potentially be utilized (see Table 4.1). Considering directed motifs may add an interesting temporal component to the patterns being investigated, by considering the order by which the stones are laid on the board and reflecting this additional information by the directed edges of the motifs.

- **Considering an Agent's Actions within Phases of the Game:** In this study, the actions of a player *(or the interactions between both players)* were investigated within the entire span of the Game. However, as we have mentioned in Section 2.1.3, a typical game of Go consists of three phases, an *Opening*, the *Middle-Game*, and the *End-of-the-Game*, with specific types of patterns – *and hence, strategies* – suitably applied within each phase *(for instance, Fuseki, Joseki, and Yose)*. Analysing an agent's actions within each phase separately could shed more light on the level of the specific skills required within that phase. Thereby, improving the overall competency awareness, and allowing for a better training by targeting precise limitations.

- **Investigating different approaches to employ the proposed framework within the Go domain:** The objective quantitative assessment provided by the fully automated competency assessment framework allows the utilization of domain knowledge in original ways. Specifically, within the Machine Learning approaches to computer Go, a major concern for the supervised training of classifiers *(e.g., Neural Networks)* has been the credit assignment problem (see Section 2.2.4); that is, the uncertainties in evaluating a best course of action upon which feedback is given for every course of action. This problem led to an increased application of alternative ways to train a classifier, such as neuro-evolution. However, rather than deciding on which actions are desirable or not – *in order to reward or punish*, a classifier using the proposed approach might be efficiently trained, for instance, to learn the actions that normally result from the application of a specific skill. This

classifier might then be used to augment the capabilities of an existing decision-making entity. Or otherwise, a group of classifiers, each trained to learn a specific individual skill, can be assembled in an ensemble to establish a decision-making entity.

Bibliography

Abbeel, P., and Andrew Y. Ng (2004). Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning*, pages, Banff, Canada.

Abramson, M., and H. Wechsler (2003). A Distributed Reinforcement Learning Approach to Pattern Inference in Go. *In Proceedings of International Conference on Machine Learning Applications*, 23–28, June.

Adam, G. (2009). No Name Go Server Archives. http://computer-go.org/pipermail/computer-go/2006-April/005343.html

Akilli, Göknur K. (2011). Games and Simulations: A New Approach in Education? *In Gaming and Simulations: Concepts, Methodologies, Tools and Applications, ed. Information Resources Management Association, USA*, pp. 150–167.

Albers, Michael J. (2005). Communication of complex information: user goals and information needs for dynamic Web information. Lawrence Erlbaum, p. 58.

Allis, Louis V. (1994). Searching for Solutions in Games and Artificial Intelligence. PhD thesis, Rijksuniversiteit Limburg; University Library, Maastricht University.

Amgoud, L., and H. Prade (2005). Handling threats, rewards and explanatory arguments in a unified setting. *International Journal of Intelligent Systems* **20**(12):1195–1218.

Andrews, R., Joachim Diederich, and Alan B. Tickle (1995). Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems* **8**(6):373–389, December.

Banbury, S., and S. Tremblay (2004). *A Cognitive Approach to Situation Awareness: Theory and Application*. Ashgate Publishing Limited.

Bassellier, G., Blaize Horner Reich, and Izak Benbasat (2001). Information Technology Competence of Business Managers: A Definition and Research Model. *Journal of Management Information Systems* **17**(4):159–182, Spring.

Beck, D., S. Ayers, J. Wen, M. Brandl, T. Pham, P. Webb, C.-C. Chang, and X. Zhou (2011). Integrative analysis of next generation sequencing for small non-coding

RNAs and transcriptional regulation in Myelodysplastic Syndromes. *BMC Medical Genomics* **4**(19).

Bennet, A., and D. Bennet (2008). The Decision-Making Process in a Complex Situation. *Handbook on Decision Support Systems 1, International Handbooks on Information Systems, Springer Berlin Heidelberg*, pp.3–20.

Blili, S., L. Raymond, and S. Rivard (1998). Impact of task uncertainty, end-user involvement, and competence on the success of end-user computing. *Information and Management* **33**(3):137–153.

Bonissone, P., R. Subbu, N. Eklund, and T. Kiehl (2006). Evolutionary Algorithms + Domain Knowledge = Real-World Evolutionary Computation. *IEEE Transactions on Evolutionary Computation* **10**(3):256–280, June.

Bouzy, B. (1996). Spatial Reasoning in the game of Go. *In Workshop on Representations and Processes in Vision and Natural Language*, ECAI.

Bouzy, B. (2003). Mathematical morphology applied to computer Go. *International Journal of Pattern Recognition and Artificial Intelligence* **17**(2):257–268.

Bouzy, B. (2004). Associating Shallow and Selective Global Tree Search with Monte Carlo for 9×9 Go. Lecture *Notes in Computer Science*, **3846**, 67–80.

Bouzy, B. (2005). Associating domain-dependent knowledge and Monte Carlo approaches within a Go program. *Information Sciences, Heuristic Search and Computer Game Playing IV, Edited by K. Chen*, **175**(4): 247–257, Elsevier.

Bouzy, B., and T. Cazenave (1997). Shared concepts between complex domains and the game of Go. *In Proceedings of the first International and Interdisciplinary Conference on Modelling and Using Context (CONTEXT-97)*, Rio de Janeiro - Brazil, February 4-6.

Bouzy, B. and T. Cazenave (2001). Computer Go: an AI oriented survey. *Artificial Intelligence*, **132**(1): 39–103, Elsevier.

Bouzy, B. and G.M.J-B. Chaslot (2005). Bayesian Generation and Integration of K-nearest-neighbor Patterns for 19×19 Go. *In G. Kendall and Simon Lucas, editors, IEEE 2005 Symposium on Computational Intelligence in Games - Essex, UK*, 176–181.

Bouzy, B., and B. Helmstetter (2003). Monte-Carlo Go Developments. *Advances in Computer Games: Many Games, Many Challenges: Proceedings of the ICGA/IFIP*

SG16 10th Advances in Computer Games Conference (ACG'10) - Graz - Styria, Austria, 159–174, November 24-27.

Breiman, L. (2001). Random Forests. *Machine Learning - Springer*, **45**(1):5-32.

Bringsjoid, S. (1998). Chess is Too Easy. *MIT's Technology Review - Massachusetts Institute of Technology* **101**(2):23–28, March-April.

Brügmann, B. (1993). Monte Carlo Go. *Technical Report, Max-Planck-Institute of Physics, München - Germany*, October 9.

Buford, John F., Gabriel Jakobson, and Lundy Lewis (2010). Peer-to-peer coupled agent systems for distributed situation management. *Information Fusion* **11**(3):233–242.

Bullen, G., and L. Sacks (2003). Towards new Modes of Decision Making—Complexity and Human factors. *Proceedings of the Complexity, Ethics and Creativity Conference, London* **19**(1):1–5.

Bump, D., et al., GNU Go. http://www.gnu.org/software/gnugo/gnugo.html

Burmeister, Jay M. (2000). Studies in Human and Computer Go: Assessing the Game of Go as a Research Domain for Cognitive Science. PhD thesis, School of Computer Science and Electrical Engineering and School of Psychology, The University of Queensland, Australia.

Burmeister, J. and J. Wiles (1995). The challenge of Go as a domain for AI research: a comparison between Go and chess. *In Proceedings of the Third Australian and New Zealand Conference on Intelligent Information Systems (ANZIIS-95)*, pp.181–186, 27 November.

Burmeister, J., and Janet Wiles (1997). AI Techniques Used in Computer Go. *In Proceedings of the Fourth Conference of the Australasian AI Society*.

Burmeister, J. M., Y. Saito, A. Yoshikawa, and J. H. Wiles (2000). Memory performance of master go players. *In van der Herick, Iisa (Ed.), Games in AI Research* **1**:271–286, Venlo, The Netherlands: Van Spijk.

Cai, X., and Donald C. Wunsch II (2007). Computer Go: A Grand Challenge to AI. *Studies in Computational Intelligence (SCI)* **63**:443–465. Springer-Verlag Berlin Heidelberg.

Campbell, M., A.J. Hoane Jr., and F-H. Hsu (2002). Deep Blue. *Artificial Intelligence*, **134**(1-2): 57–83.

Catania, B., A. Maddalena, and M. Mazza (2005). PSYCHO: a prototype system for pattern management. *Proceedings of the 31st International Conference on Very Large Data-Bases (VLDB'05), Trondheim – Norway*, pp. 1346–1349.

Cazenave, T. (2002). Metarules to improve tactical Go knowledge. *Information Sciences* **154**:173–188, Elsevier.

Cazenave, T. and B. Helmstetter (2005). Combining Tactical Search and Monte-Carlo in the Game of Go. *In Proceedings of the IEEE Symposium on Computational Intelligence and Games*, 171–175.

Computer Go Server (CGOS). [Online]. Available: http://cgos.boardspace.net/.

Chandana, S., and Henry Leung (2010). A System of Systems Approach to Disaster Management. *Communications Magazine*, *IEEE Communications Society* **48**(3):138–145, March.

Chaslot, G.M.J-B., M.H.M. Winands, H.J. van der Herik, J.W.H.M. Uiterwijk, and B. Bouzy (2008). Progressive Strategies for Monte-Carlo Tree Search. *New Mathematics and Natural Computation (NMNC)*, **4**(3): 343–357, World Scientific Publishing Co. Pte. Ltd.

Chaslot, G., C. Fiter, J.B. Hoock, A. Rimmel, and O. Teytaud (2010). Adding expert knowledge and exploration in Monte-Carlo Tree Search. *Advances in Computer Games*, 1–13, Springer.

Chellapilla, K. and D. Fogel (1999). Evolving Neural Networks to Play Checkers without Relying on Expert Knowledge. *IEEE Transactions on Neural Networks*, **10**(6): 1382–1391, November.

Chen, Keh-Hsun, Dawei Du and Peigang Zhang (2009). Monte-Carlo Tree Search and Computer Go. *Advances in Information and Intelligent Systems - Studies in Computational Intelligence*, **251**, 201–225.

Chikun, C. (1997). *Go: A Complete Introduction to the Game*. Kiseido Publishing Company, September.

Clerveaux, V., and B. Spence, and T. Katada (2010). Promoting disaster awareness in multicultural societies: the DAG approach. *Disaster Prevention and Management* **19**(2):199–218.

Connors, M.H., and B.D. Burns, and G. Campitelli (2011). Expertise in Complex Decision Making: The Role of Search in Chess 70 Years After de Groot. *Cognitive Science* **35**:1567–1579.

Cook, M., J. Noyes, and Y. Masakowski (2007). Decision-making in Complex Environments. Ashgate Publishing Company, March.

Coulom, R. (2006). Efficient selectivity and backup operators in Monte-Carlo tree search. *In Proceedings of the 5th international conference on Computers and Games (CG'06)*, Springer-Verlag Berlin, Heidelberg.

Coulom, R. (2007). Computing Elo Ratings of Move Patterns in the Game of Go. *ICGA journal*, **30**(4): 198–208.

Coulom, R. (2009). The Monte-Carlo Revolution in Go. *Japanese-French Frontiers of Science Symposium (JFFoS'2008)*, January.

Davies, L., and Ursula Gather (1993). The Identification of Multiple Outliers. *Journal of the American Statistical Association*, 88(423):782-792, Sep. 1993.

de Groot, A. (1946). *Thought and Choice in Chess*. In Dutch; English edition in 1965, The Hague, Mouton Publishers.

Duda, R.O., P.E. Hart, and D.G. Stork (2001). *Pattern Classification*. New York: John Wiley.

Durso, Francis T., and Arathi Sethumadhavan (2008). Situation Awareness: Understanding Dynamic Environments. *Human Factors: The Journal of the Human Factors and Ergonomics Society* **50**:442–448, June.

El-Fiqi, H., Eleni Petraki, and Hussein A. Abbass (2011). A Computational Linguistic Approach for the Identification of Translator Stylometry using Arabic-English Text. *IEEE International Conference on Fuzzy Systems, Taipei – Taiwan*, pp. 2039–2045, June 27-30.

Endsley, Mica R. (1995). Toward a Theory of Situation Awareness in Dynamic Systems. *Human Factors* **37**(1):32–64, March.

Endsley, Mica R. (2000). Theoretical Underpinnings of Situation Awareness: A Critical Review. In M. R. Endsley & D. J. Garland (Eds.), *Situation Awareness: Analysis and Measurement* (pp. 3–32). Lawrence Erlbaum Associates, Inc - CRC Press.

Endsley, Mica R., and Daniel J. Garland (2000). *Situation Awareness: Analysis and Measurement*. Lawrence Erlbaum Associates, Inc - CRC Press.

Enzenberger, M., M. Müller, B. Arneson and R. Segal (2010). 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 - Special issue on Monte Carlo Techniques and Computer Go*, **2**(4): 259–270.

Ernandes, M. (2005). Artificial intelligence & games: Should computational psychology be revalued? *Topoi*, **24**(2):229–242, Springer.

Fabian, N. (2008). *Machine Learning of Human Behavior in Interactive Games*. MSc thesis, The University of New Mexico, Albuquerque, New Mexico, December.

Farrington-Darby, T., and John R. Wilson (2006). The Nature of Expertise: A Review. *Applied Ergonomics* **37**:17–32, Elsevier.

Fauconnier, C., and G. Haesbroeck. Outliers detection with the minimum covariance determinant estimator in practice. *Statistical Methodology*, 6(4):363-379, 2009.

Feng, Y.-H., Teck-Hou Teng, Ah-Hwee Tan (2009). Modelling Situation Awareness for Context-Aware Decision Support. *Expert Systems with Applications* **36**:455–463.

Fogel, D., T.J. Hays, S.L. Hahn, and J. Quon (2004). A Self-Learning Evolutionary Chess Program. *In Proceeding of the IEEE*, **92**(12): 1947–1954, December.

Frantz, R. (2003). Herbert Simon. Artificial intelligence as a framework for understanding intuition. *Journal of Economic Psychology- Elsevier* **24**(2):265–277.

Furse, E., and R. Nicolson (1993). Perception and Experience in Problem Solving. *Cognitive Modelling* **13**:181–186.

Garcia-Retamero, R., and U. Hoffrage (2006). How causal knowledge simplifies decision-making. *Minds and Machines* **16**:365–380, Springer-Verlag.

Gelly, S., and D. Silver (2007). Combining online and offline knowledge in UCT. In Proceedings of the 24th international conference on Machine learning (ICML'07), ACM New York USA.

Gelly, S., and D. Silver (2008). Achieving master level play in 9x9 computer Go. *In Proceedings of the 23rd National Conference on Artificial intelligence (AAAI'08),* pp. 1537–1540.

Gelly, S., Y. Wang, R. Munos, and O. Teytaud (2006). Modification of UCT with patterns in Monte-Carlo Go. *Research Report - Institut National de Recherche en Informatique et en Automatique (INRIA)*, November.

Ghoneim, A. S. (2008). Local-Global Coupling in Strategy Games Extracting Signatures and Unfolding Dynamics. M.Sc. thesis, School of Information Technology and Electrical Engineering, University of New South Wales at the Australian Defence Force Academy (UNSW@ADFA), Canberra - Australia.

Ghoneim, A. S., and D. L. Essam (2012). A Methodology for Revealing and Monitoring the Strategies Played by Neural Networks in Mind Games. Accepted by the 2012 International Joint Conference on Neural Networks (IJCNN), at the 2012 IEEE World Congress on Computational Intelligence (WCCI 2012), Brisbane – Australia, June 10-15.

Ghoneim, A. S., Hussein A. Abbass, Michael Barlow (2008). Characterizing Game Dynamics in Two-Player Strategy Games using Network Motifs. *IEEE Transactions on Systems, man, and Cybernetics—Part B: Cybernetics* **38**(3):682–690, June.

Ghoneim, A. S., D. L. Essam, and H. A. Abbass (2011). Competency Awareness in Strategic Decision Making. In Proceedings of the IEEE First International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support (CogSIMA), pp. 106–109, 22–24 Feb. 2011, Miami Beach, FL, USA. IEEE Press.

Ghoneim, A. S., D. L. Essam, and H. A. Abbass (2011). On Computations and Strategies for Real and Artificial Systems. In Advances in Artificial Life, ECAL 2011: Proceedings of the Eleventh European Conference on the Synthesis and Simulation of Living Systems, edited by Tom Lenaerts, Mario Giacobini, Hugues Bersini, Paul Bourgine, Marco Dorigo and René Doursat, pp. 260–267, 8–12 August 2011, Paris, France. MIT Press.

Ginsberg, M.L. (1999). GIB: Steps toward an Expert-Level Bridge-Playing Program. *International Joint Conference on Artificial Intelligence*, **16**, 584–593.

Glimcher, Paul W., Colin F. Camerer, Ernst Fehr, and Russell A. Poldrack (2009). Neuroeconomics: Decision making and the brain. Elsevier Inc.

Gobet, F. (2005). Chunking Models of Expertise: Implications for Education. *Applied Cognitive Psychology* **19**:183–204.

Gobet, F. (2006). Adriaan De Groot: Marriage of Two Passions. *International Computer Games Association (ICGA) Journal* **29**(4):236–243.

Gobet, F., A.J. de Voogt, and J. Retschitzki (2004). *Moves in mind: The psychology of board games*. Psychology Press.

Govier, T. (1989). Critical Thinking as Argument Analysis? *Argumentation* III:115-126.

Graham, John R., Jack A. Naglieri, and Irving B. Weiner (2003). *Handbook of Psychology: Assessment Psychology (Volume 10)*. Wiley, January.

Groth-Marnat, G. (2009). *Handbook of psychological assessment*. Wiley.

Harré, M., Terry Bossomaier, and Allan Snyder (2011). The Development of Human Expertise in a Complex Environment. *Minds and Machines* **21**(3):449–464, April, Springer-Verlag.

Harré, M.S., T. Bossomaier, A. Gillett, and A. Snyder (2011). The aggregate complexity of decisions in the game of Go. *The European Physical Journal B*, **80**(4):555–563, Springer-Verlag.

Hawkins, D. (1980). *Identification of Outliers*. Chapman and Hall.

Hmelo-Silver, Cindy E., and Merav Green Pfeffer (2004). Comparing Expert and Novice Understanding of a Complex System from a Perspective of Structures, Behaviours, and Functions. *Cognitive Science* **28**:127–138, Elsevier.

Ho, Y.-C., and D. Pepyne (2002). Simple Explanation of the No Free Lunch Theorem of Optimization. *Cybernetics and Systems Analysis* **38**(2):292-298.

Hoffman, Robert R. (1998). How can expertise be defined? Implications of research from cognitive psychology. *In R. Williams, W. Faulkner & J. Fleck (Eds.) Exploring expertise*, pp. 81–100.

Hollosi, A. (2009). SGF File Format FF[4], The official specification of the Smart Game Format - File Format version 4. http://www.red-bean.com/sgf/index.html

Hoock, J.B., C.S. Lee, A. Rimmel, F. Teytaud, M.H. Wang, and O. Teytaud (2010). Intelligent agents for the game of Go. *IEEE Computational Intelligence Magazine*, **5**(4):28–42, IEEE.

Hoppenbrouwers, S., and P.J.F. Lucas (2009). Attacking the knowledge acquisition bottleneck through Games-For-Modelling. *Proceedings of the AI and Games Symposium, at the Artificial Intelligence and Simulation of Behaviour (AISB) Convention, Heriot-Watt University - Edinburgh - Scotland*, pp. 81–86, 6-9 April.

Hsu, F. (2002). *Behind Deep Blue: Building the Computer that Defeated the World Chess Champion*. Princeton, NJ: Princeton Univ. Press.

Iacovides, I. (2009). Exploring the link between player involvement and learning within digital games. *In Proceedings of the 23rd BCS HCI 09 Conf. on People and Computers*, Swinton, UK.

Jakobson, G., L. Lewis, C.J. Matheus, M.M. Kokar, and J. Buford (2005). Overview of Situation Management at SIMA 2005. *The Workshop on Situation Management (SIMA) at the IEEE Military Communications Conference (MILCOM 2005)* **3**:1630–1636, 17-20 October, Atlantic City, NJ.

Jakobson, G., John Buford, and Lundy Lewis (2007). Situation Management: Basic Concepts and Approaches. *In Vasily V. Popovich, Manfred Schrenk, Kyrill V. Korolenko (Eds.), Proceedings of the Third International Workshop on Information Fusion and Geographical Information Systems (IF&GIS 07), St. Petersburg, Russia, May 27-29. Lecture Notes in Geoinformation and Cartography*, pp. 18–33, Springer.

Jin, Y. (2004). Knowledge Incorporation in Evolutionary Computation. *Studies in Fuzziness and Soft Computing* **167**, Springer.

Johnson, R. (1992). *Applied Multivariate Statistical Analysis*. Prentice Hall.

Kiin, N., and K. Kiin (1989). The Japanese Rules of Go. Online: http://www.cs.cmu.edu/wjh/go/rules/Japanese.html, April. (Originally in Japanese: Nihon Igo Kiyaku). Translated by J. Davies, diagrams by J. Cano, reformatted, adapted, and edited by F. Hansen

Kim, J. (1981). Causes as Explanations: A Critique. *Theory and Decision* **13**:293–309, D. Reidel Publishing Co., Dordrecht, Holland, and Boston, USA.

Kim, K.-J., and S.-B. Cho (2007). Evolutionary Algorithms for Board Game Players with Domain Knowledge. *Studies in Computational Intelligence (SCI)* **17**:71–89.

Kim, T.H., and J.A. Nisbett and D.C. Wunsch (2009). Robotic Go: Exploring a Different Perspective on Human-Computer Interaction with the Game of Go. *In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics (SMC'09), San Antonio, TX, USA – October*, pp. 2439–2444.

Kittler, J., M. Hatef, R. Duin, and J. Matas (1998). On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226-239, Mar. 1998.

Klein, Gary A., and Robert R. Hoffman (1992). Seeing the Invisible: Perceptual–Cognitive Aspects of Expertise. *In M. Rabinowitz (Ed.), Cognitive Science Foundations of Instruction*, pp. 203–226, Mahwah, NJ: Erlbaum.

Klinger, T. (2001). Adversarial Reasoning: A Logical Approach for Computer Go. PhD thesis, New York University.

Kocsis, L., and C. Szepesvári (2006). Bandit based Monte-Carlo Planning. *In Proceedings of the 17th European Conference on Machine Learning Berlin, Germany, September 18-22, Lecture Notes in Computer Science* **4212**: 282–293.

Konidaris, G., D. Shell, and N. Oren (2002). Evolving Neural Networks for the Capture Game. *In Proceedings of the SAICSIT Postgraduate Symposium - Port Elizabeth, South Africa*.

Kumar, S.R. (2003). Caching pattern and method for caching in an object-oriented programming environment. *US Patent 6,651,140 – Google Patents*, November.

Lado, A.A., and M.C. Wilson (1994). Human Resource Systems and Sustained Competitive Advantage: A Competency-Based Perspective. *Academy of Management Review* **19**(4):699–727, Academy of Management.

Lai, D. (2004). Learning from the Stones: A Go Approach to Mastering China's Strategic Concept, Shi. *Technical report, The Strategic Studies Institute of the US Army War College, Carlisle, PA*, May.

Lee, C.S., and M. Müller and O. Teytaud (2010). Special Issue on Monte Carlo Techniques and Computer Go. *IEEE Transactions on Computational Intelligence and AI in Games* **2**(4):225–228.

Lee, C.S., M.H. Wang, S.J. Yen, Y.J. Chen, C.W. Chou, G. Chaslot, J.B. Hoock, A. Rimmel, and Doghmen, H. (2010). An ontology-based fuzzy inference system for computer Go applications. *International Journal of Fuzzy Systems* **12**(2):103–115.

Lewis, D. (1986). Causal Explanation. *Philosophical Papers* Vol. **II**, Oxford University Press.

Lichtenstein, D., and M. Sipser (1980) Go is polynomial-space hard. *Journal of the ACM (JACM)* **27**(2):393–401.

Liew, A., and Sundaram, D. (2005). Complex Decision Making Processes: their Modelling and Support. *In Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS'05)*, **3**: 88c.

Linhares, A. (2005). An Active Symbols Theory of Chess Intuition. *Minds and Machines - Springer Netherlands* **15**(2):131–181.

Loh, S., and S. H. Soon (2006). Comparing computer and traditional games using game design patterns. *In Proceedings of the 2006 international conference on Game research and development – CyberGames'06*, pp. 237–241, Murdoch University- Perth, Australia.

Loo, R. (2003). A multi-level causal model for best practices in project management. *Benchmarking: An International Journal (BIJ)* **10**(1):29–36.

Lubberts, A., and R. Miikkulainen (2001). Co-Evolving a Go-Playing Neural Network. *In Proceedings of the GECCO-01 Workshop on Coevolution: Turning Adaptive Algorithms upon Themselves – San Francisco, California, USA*, 14–19.

Lucia, A. D., and R. Lepsinger (1999). *The Art and Science of Competency Models*. Jossey-Bass / Pfeiffer; San Francisco.

Luger, G. F., and W. A. Stubblefield (1998). *Artificial Intelligence: Strategies and Structures for Complex Problem Solving*. Addison Wesley Longman.

Luther, M., Bernd Mrohs, Matthias Wagner, Stephan Steglich, and Wolfgang Kellerer (2005). Situational Reasoning – A Practical OWL Use Case. *In Proceedings of the Autonomous Decentralized Systems (ISADS)*, pp. 461–468, IEEE.

Malone, T. (1981). Toward a theory of intrinsically motivating instruction. *Cognitive Science, Elsevier* **5**:333–369.

Mańdziuk, J. (2007). Computational Intelligence in Mind Games. *In Challenges for Computational Intelligence, Studies in Computational Intelligence (SCI), Wlodzislaw Duch and Jacek Mandziuk, Ed.*, **63**, 407–442, Springer Berlin / Heidelberg.

Mańdziuk, J. (2008). Some thoughts on using Computational Intelligence methods in classical mind board games. *IEEE World Congress on Computational Intelligence - IEEE International Joint Conference on Neural Networks, 2008. (IJCNN'08)*, pp. 4002–4008.

Mayer, H. A. (2007). Board Representations for Neural Go Players Learning by Temporal Difference. *In IEEE Symposium on Computational Intelligence and Games (CIG 2007)*, 183–188.

Mayer, H. A., and P. Maier (2005). Coevolution of Neural Go Players in a Cultural Environment. *In Proceedings of the 2005 IEEE Congress on Evolutionary Computation (CEC'05)* **2**:1017–1024, September.

McClelland, D.C. (1973). Testing for competence rather than for "intelligence." *American Psychologist* **28**(1):1–14, American Psychological Association, January.

Meijer, A. B., and H. Koppelaar (2001). A Learning Architecture for the Game of Go. *In Proceedings of the 2nd Annual European Conference on Simulation and AI in Computer Games (GAME-ON-01).*

Mirabile, R.J. (1997). Everything you wanted to know about competency modeling. *Training and development* **51**(8):73–77.

Moriarty, D. E., and R. Miikkulainen (1995). Discovering Complex Othello Strategies through Evolutionary Neural Networks. *Connection Science* **7**:195–209.

Moriarty, D. E., and R. Miikkulainen (1997). Forming Neural Networks through Efficient and Adaptive Coevolution. *Evolutionary Computation* **5**(4):373–399.

Müller, M. (1999). Decomposition search: A combinatorial games approach to game tree search, with applications to solving Go endgames. *In Proceedings of the 16th International Joint Conference on Artificial Intelligence (IJCAI'99)* **1**:578–583.

Müller, M. (2000). Not like other games—Why tree search in Go is different. *In Proceedings of the 5th Joint Conference on Information Sciences (JCIS).*

Müller, M. (2001) Review: Computer Go 1984–2000. *In Marsland, T. and Frank, I. (eds), Computers and Games 2000. Lecture Notes in Computer Science - Springer-Verlag* **2063**:426–435.

Müller, M. (2002). Computer Go. *Artificial Intelligence* **134**(1-2): 145–179.

Müller, M. (2003). Conditional combinatorial games and their application to analysing capturing races in Go. *Information Sciences*, **154**(3): 189–202, Elsevier, September.

Munro, M. C., Sid L. Huff, Barbara L. Marcolin, and Deborah R. Compeau (1997). Understanding and measuring user competence. *Information and Management* **33**(1):45–57.

Murray, T. (1997). Expanding the Knowledge Acquisition Bottleneck for Intelligent Tutoring Systems. *In the International Journal of Artificial Intelligence in Education (IJAIED) - Special Issue on Authoring Systems for Intelligent Tutoring Systems* **8**(3).

National Computational Infrastructure (NCI) National Facility: Sun Constellation Cluster, Vayu: System Details (2011 Feb. 9). [Online]. Available: https://nf.apac.edu.au/facilities/vayu/hardware.php.

Newell, A., and Herbert A. Simon (1972). *Human Problem Solving*. Prentice-Hall, Englewood Cliffs, NJ.

Newmann, M. (2010). *Networks: An Introduction*. New York: Oxford University Press.

Padilla, J. J., Andres Sousa-Poza, Arturo Tejada, and Samuel Kovacic (2007). Towards a Diagnostic Framework for Understanding Complex Situations. *In Proceedings of the 7th International Conference on Complex Systems, Boston – Massachusetts*.

Parry, S. B. (1996). The quest for competences: competency studies can help you make HR decision, but the results are only as good as the study. *Training* **33**:48–56.

Pavlas, D., K. Heyne, W. Bedwell, E. Lazzara, and E. Salas (2010). Game-based Learning: The Impact of Flow State and Videogame Self-efficacy. *In Proceedings of the Human Factors and Ergonomics Society Annual Meeting* **54**(28):2398–2402.

Pew, R. W. (2000). The State of Situation Awareness Measurement: Heading Toward the Next Century. In M. R. Endsley & D. J. Garland (Eds.), *Situation*

Awareness: Analysis and Measurement (pp. 33–47). Lawrence Erlbaum Associates, Inc - CRC Press.

Phillips-Wren, G., and Jain, L. (2006) Artificial Intelligence for Decision Making. *Knowledge-Based Intelligent Information and Engineering Systems, Lecture Notes in Computer Science, Springer Berlin / Heidelberg* **4252**: 531–536.

Pollack, J. B., and Alan D. Blair (1998). Co-Evolution in the Successful Learning of Backgammon Strategy. *Machine Learning*, **32**, 225–240.

Raiffa, H. (1968). *Decision Analysis: Introductory lectures on choices under uncertainty*. MA: Addison-Wesley.

Raiko, T. (2005). Nonlinear relational Markov networks with an application to the game of Go. *In Proceedings of the 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications (ICANN'05) - Lecture notes in computer science, Springer-Verlag - Volume Part II*, 989–996.

Rangel, A., C. Camerer, P.R. Montague (2008). A framework for studying the neurobiology of value-based decision making. *Nature Reviews Neuroscience* **9**(7): 545–556.

Rasskin-Gutman, D. (2009). *Chess metaphors: artificial intelligence and the human mind*. The MIT Press.

Reitman, J.S. (1976). Skilled perception in Go: Deducing memory structures from inter-response times. *Cognitive psychology* **8**(3):336–356, Elsevier.

Remus, H. (1962). Simulation of a Learning Machine for Playing Go. *In Information Processing: Proceedings of the IFIP Congress*, 428–432, North-Holland Publishing Company.

Richards, N., David Moriarty, and Risto Miikkulainen (1998). Evolving Neural Networks to Play Go. *Applied Intelligence*, **8**, 85–96.

Rimmel, A., O. Teytaud, C.S. Lee, S.J. Yen, M.H. Wang, and S.R. Tsai (2010). Current Frontiers in Computer Go. *IEEE Transactions on Computational Intelligence and AI in Games* **2**(4):229–238.

Rosin, C. D., and R. K. Belew (1997). New Methods for Competitive Coevolution. *Evolutionary Computation*, **5**(1): 1–29.

Ross, P.E. (2006). The expert mind. Scientific *American - Nature Publishing Group* **295**(2):64–71.

Rousseeuw, P. J. (1985). Multivariate Estimation with High Breakdown Point. *Mathematical Statistics and Applications* **8**:283–297, Reidel, Dordrecht.

Rousseeuw, P. J., and K. Van Driessen (1999). A Fast Algorithm for the Minimum Covariance Determinant Estimator. *Technometrics* **41**(3):212–223.

Sanfey, A. G. (2007). Social Decision-Making: Insights from Game Theory and Neuroscience. *Science* **318**: 598–602.

Schaeffer, J., N. Burch, Y. Björnsson, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen (2007). Checkers is Solved. *Science*, July.

Sei, S., and T. Kawashima (2000). A solution of go on 4x4 board by game tree search program. I*n The 4th Game Informatics Group Meeting in IPS Japan*, 69–76. Fujitsu Social Science Laboratory. (in Japanese) Translation available at http://homepage1.nifty.com/Ike/katsunari/paper/4x4e.txt.

Shannon, C. E. (1950). Programming a computer for playing chess. *Philosophical Magazine Series 7* **41**(314):256–275.

Shapiro, S. C. (1992). Artificial Intelligence. In S. C. Shapiro (ed.), Encyclopedia of Artificial Intelligence, New York, NY: Wiley, pp. 54–57.

Sharma, M., Michael Holmes, Juan Santamaria, Arya Irani, Charles Isbell, and Ashwin Ram (2007). Transfer learning in realtime strategy games using hybrid CBR/RL. *In Proceedings of the Twentieth International Joint Conference on Artificial Intelligence (IJCAI'07) – Hyderabad-India*, pp. 1041–1046.

Sheppard, B. (2006). Efficient Control of Selective Simulations. *Computers and Games - Lecture Notes in Computer Science*, **3846**: 1–20, Springer.

Shippmann, J.S., R.A. Ash, M. Batjtsta, L. Carr, L.D. Eyde, B. Hesketh, J. Kehoe, K. Pearlman, E.P. Prien, and J.I. Sanchez (2000). The practice of competency modeling. *Personnel Psychology* **53**(3):703–740, Wiley Online Library.

Silver, D., and J. Veness (2010). Monte-Carlo Planning in Large POMDPs. *In Proceedings of the Conference on Neural Information Processing Systems*, December.

Silver, D., Richard Sutton, and Martin Müller (2007). Reinforcement Learning of Local Shape in the Game of Go. *In 20th International Conference on Artificial Intelligence*, 1053–1058.

Simon, H.A. (1956). Rational choice and the structure of the environment. *Psychological review* **63**(2):129–138.

Simon, H. A., and W. G. Chase (1973). Skill in Chess: Experiments with chess-playing tasks and computer simulation of skilled performance throw light on some human perceptual and memory processes. *American Scientist* **61**(4):394–403, July-August.

Squire, K. (2006). From content to context: Videogames as designed experience. *Educational researcher* **35**(8):19–29.

Stanley, K. O., and R. Miikkulainen (2004). Evolving a Roving Eye for Go. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO), New York, NY*, Springer-Verlag.

Stoker, J.I., and B.I.J.M. Van der Heijden (2001). Competence Development and Appraisal in Organizations. *Journal of Career Development* **28**(2):97–113.

Sutskever, I., and V. Nair (2008). Mimicking Go Experts with Convolutional Neural Networks. *Artificial Neural Networks (ICANN 2008) - Ed. Kurková, Véra and Neruda, Roman and Koutník, Jan - Lecture Notes in Computer Science - Springer Berlin / Heidelberg* **5164**:101–110.

Sutton, R.S. (1988). Learning to predict by the methods of temporal differences. *Machine learning*, **3**(1): 9–44, Springer.

Sutton, R. S., and A. G. Barto (1998). *Reinforcement Learning: An Introduction*. MIT Press.

Tan, J., and G. Biswas (2007). Simulation-Based Game Learning Environments: Building and Sustaining a Fish Tank. *In Proceedings of the First IEEE International Workshop on Digital Game and Intelligent Toy Enhanced Learning (DIGITEL '07)*, pp.73–80, 26-28 March.

Tan, K. B., Jason Teo, and Patricia Anthony (2010). Multi-Objective Evolution of Neural Go Players. *In Proceedings of the IEEE International Conference on Digital game and Intelligent Toy Enhanced Learning*, 46–53.

Tesauro, G. (1989). Neurogammon Wins Computer Olympiad. *Neural Computation*, **1**, 321–323.

Tilley, J. J. (2004). Justifying Reasons, Motivating Reasons, and Agent Relativism in Ethics. *Philosophical Studies: An International Journal for Philosophy in the Analytic Tradition* **118**(3):373–399, April, Springer.

Tokutsu, S., Kei Okada, and Masayuki Inaba (2009). Environment Situation Reasoning Integrating Human Recognition and Life Sound Recognition using DBN. *In proceeding of the 18th IEEE international Symposium on Robot and Human Interactive Communication, Toyama – Japan*, pp. 744–750, Sept. 27 – Oct. 2.

Torkzadeh, G., and J. Lee (2003). Measures of perceived end-user computing skills. *Information and Management* **40**(7):607–615.

Tromp, J., and G. Farnebäck (2007). Combinatorics of go. *Computers and Games - Springer* **4630**:84–99.

Unluturk, M. S., Sevcan Unluturk, Fikret Pazir, and Firoozeh Abdollahi (2011). Capillary Dynamolysis Image Discrimination Using Neural Networks. *Journal of Information Technology and Software Engineering* **1**(101):1–7, November.

Van der Broeck, G., Kurt Driessens, and Jan Ramon (2009). Monte-Carlo Tree Search in Poker Using Expected Reward Distributions. *In Proceedings of the 1st Asian Conference on Machine Learning (ACML'09): Advances in Machine Learning*.

Van den Herik, H. J., J. W. H. M. Uiterwijk, and J. van Rijswijck (2002). Games Solved: Now and in the Future. *Artificial Intelligence*, **134**(1-2): 277–311.

Van der Werf, E. C. D. (2004). AI Techniques for the Game of Go. PhD thesis, Universiteit Maastricht, Maastricht, The Netherlands.

Van der Werf, E. C. D., H. J. van den Herik, and J. W. H. M. Uiterwijk (2003). Solving Go on Small Boards. *ICGA Journal* **26**(2): 92–107, June.

Van Lishout, F., Guillaume Chaslot, and Jos W.H.M. Uiterwijk (2007). Monte-Carlo Tree Search in Backgammon. *Computer Games Workshop, Amsterdam – The Netherlands*, 175–184, June 15-17.

von Winterfeldt, D., and W. Edwards (1986). *Decision Analysis and Behavioural Research*. New York: Cambridge University Press.

Wagner, A. (1999). Causality in Complex Systems. *Biology and Philosophy* **14**:83–101, Kluwer Academic Publishers, Printed in the Netherlands.

Wagner, C. (2006). Breaking the knowledge acquisition bottleneck through conversational knowledge management. *Information Resources Management Journal*, **19**(1): 70–83.

Walton, D. N. (1990). What is Reasoning? What Is an Argument? *The Journal of Philosophy* **87**(8):399–419, August.

Weinstein, B. D. (1993). What is an Expert? *Theoretical Medicine* **14**:57–73.

Wernicke, S., and F. Rasche (2006). FANMOD: a Tool for Fast Network Motif Detection. *Bioinformatics* **22**(9):1152–1153.

Willmott, S., J. Richardson, A. Bundy, and J. Levine (2001). Applying adversarial planning techniques to Go. *Theoretical Computer Science - Elsevier* **252**(1-2):45–82.

Wolfe, D. (2002). Go endgames are PSPACE-hard. *More Games of No Chance* **42**:125–136.

Woodruffe, C. (1993). What is meant by a competency? *Leadership & Organization Development Journal* **14**(1):29–36.

Wolpert, D., and W. MacReady (1997). No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation* **1**(1):67–82.

Wu, L., and P. Baldi (2008). Learning to play Go using recursive neural networks. *Neural Networks* **21**(9):1392–1400.

Yau, S. S., and J. Liu (2006). Hierarchical Situation Modelling and Reasoning for Pervasive Computing. *The Fourth IEEE Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS) and the Second International Workshop on Collaborative Computing, Integration, and Assurance (WCCIA)*.

Yoon, C.Y. (2009). Measures of perceived end-user computing competency in an organizational computing environment. *Knowledge-Based Systems* **22**(6):471–476, Elsevier.

Yoshikawa, A., and Takuya Kojima, and Yasuki Saito (1999). Relations between Skill and the Use of Terms: An Analysis of Protocols of the Game of Go. *In van den*

Herik, H. and Iida, Hiroyuki (Eds.) Computers and Games, Lecture Notes in Computer Science **1558**:282–299, Springer Berlin / Heidelberg.

Yoshimoto, H., K., T. Kaneko Yoshizoe, A. Kishimoto, and K. Taura (2006). Monte Carlo Go has a Way to Go. *In Proceedings of the 21st National Conference on Artificial Intelligence,* **21(**2): 1070–1075, AAAI Press.

Ziebart, B. D., Andrew Maas, J. Andrew Bagnell, and Anind K. Dey (2008). Maximum Entropy Inverse Reinforcement Learning. In *Proceedings of the Association for the Advancement of Artificial Intelligence (AAAI)*, pp. 1433–1438, July.

Zobrist, Albert L. (1970). Feature Extraction and Representation for Pattern Recognition and the Game of Go. PhD thesis, The University of Wisconsin, Madison.