Publication V

Esa A. Seuranen. 2007. Introducing playing style to computer go. In: Jaap van den Herik, Jos Uiterwijk, Mark Winands, and Maarten Schadd (editors). Proceedings of the Computer Games Workshop 2007 (CGW 2007). Amsterdam, The Netherlands. 15-17 June 2007. Maastricht, The Netherlands. Maastricht University. MICC Technical Report Series, 07-06, pages 81-91.

Introducing Playing Style to Computer Go

Esa A. Seuranen

Department of Electrical and Communications Engineering
Helsinki University of Technology
P.O. Box 3000, 02015 TKK, Finland
esa.seuranen@tkk.fi

Abstract. We give a brief overview of go and the methods applied in computer go. We discuss the weaknesses in the current approaches and propose a design, which is aimed to overcome some of these shortcomings. The main idea of the design is to handle the game with subgames. These subgames consist of a part of the whole board and they have a list of local purposes. The next move is chosen from the subgames according to a *playing style*, which provides values for global goals. The moves which strive towards the best outcome (in terms of global goals weighted with the style) are preferred.

1 Introduction

Go is a 4000 year old two-player board game originating from China. For the artificial-intelligence aspect go is quite interesting, because it is practically the only classical complete-information board game in which the best computer players are defeated by average human players.[1]

In this paper we discuss different aspects of go and computer go in order to state our assumptions about the game. Based on these assumptions we propose a design for a computer go player, *agent*, which should be able to overcome some weaknesses in the current approaches.

The paper is outlined as follows. In Section 2 we give an introduction to go. In Section 3 we survey the main ideas used in computer go. We introduce the concept of playing style in Section 4 and propose a design for using it in Section 5. The conclusions are given in Section 6.

2 Go

We will go over the basics briefly, that is, describe the (simplified Chinese) rules of the game, discuss the hardness of the game and emphasize the important

[1] In chess and Chinese chess (Xiang Qi) computers have reached already the level of best human players. In 2005 the Japan Shogi Association told the professional shogi players not to play against computer players publicly without a permission, so apparently the chance of defeat is quite real (http://en.wikipedia.org/wiki/Shogi, [14.5.2007]).

aspects of playing the game skillfully. For information about the world of go the reader is referred to [5].

A go game starts from an empty board, 19×19 intersections being the most common size. Black and white players make *moves* alternatively, i.e., place a stone of their own color on an empty intersection on the board. Placed stones remain on the board to the end of the game, unless they are captured. A set of horizontally or vertically adjacent stones of the same color, *block*, is captured if it has no adjacent empty intersections, *liberties*. A player is not allowed to place a stone on the last liberty of any of his blocks, unless he captures a group of opposing color by doing so. A move which repeats any previous board situation is illegal. These situations are referred as *kos*.[2] The game ends when both players pass, after which *dead* blocks (stones which could be captured even if they were defended) are removed. The player who has more stones and surrounded empty intersections on the board is the winner.

The rank scale of go is in Figure 1. Roughly speaking, a player is considered to be a beginner for 30–11 kyu range and an average for 11–1 kyu range. Higher dan ranks are considered to be quite strong amateur players. Professional players (indicated with *pro*) are a chapter of their own, as the strongest amateur players are considered to be similar strength to the weakest professional players. *Kami no Itte* refers to "Hand of God", i.e., perfect play. It is more or less generally believed that the difference between the best professional players and perfect play is 3–4 stones [32]. The rank difference between players gives directly the amount of handicap stones, which the weaker player should place on the board in order for both players to have equal chance of winning. For professional levels three ranks equals one stone.

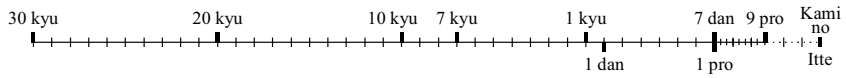

Fig. 1. Ranks in go.

In Figure 2 the rank development of European players (who have played in at least 50 tournaments and whose first rank in the system [33] was 20 kyu) is displayed over the course of years they have participated in tournaments. The triangle line is the average rank development of the players.[3] The average rank of all European tournament players is around 7 kyu.

[2] Every now and then a ko fight occurs, where players have to play threatening moves somewhere else before they can return to play a local move. The player who runs out sufficiently big threatening moves first will lose the ko fight and suffer a loss (at least locally).

[3] It is assumed that players' ranks are unchanged after their last tournament appearance.

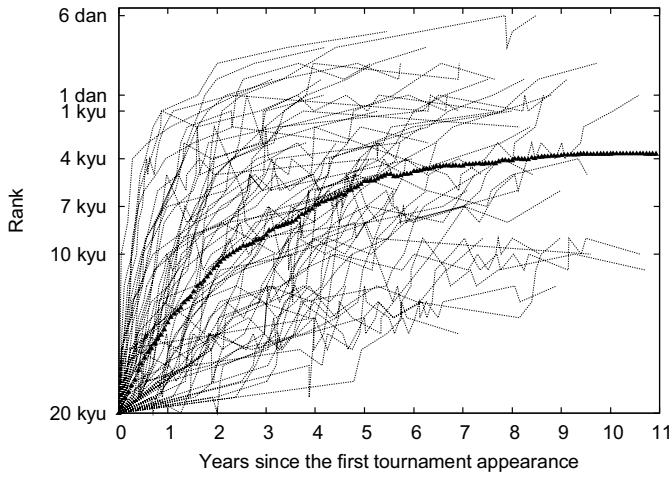

Fig. 2. Development of some human go players.

It can be said that there are four aspects in go, which make the game challenging. The first one is the difficulty of evaluating the game situation, i.e., who is winning. The second aspect is the difficulty of deciding, what should be done. The third one is about making efficient moves, i.e., finding the best local move. The fourth aspect is timing–making the right moves at the right time.

2.1 Evaluating a Game Situation

It is quite hard to name precisely the aspects that are relevant to a situation evaluation. We shall use here three terms–*territory*, *influence* and *thickness*–as the basis. The go literature supports this view, although the used terminology may vary to some degree. A situation evaluation is a combination of these three concepts.

Territory is the amount of points the player can expect to have in the end, i.e., the number of player's stones and surrounded intersections. The player's influence represents both potential of making territory and preventing his opponent from making territory. The thickness refers to the safety and stability of player's *groups* (nearby blocks of the same color). For instance, if a player has many weak groups he will have to avoid ko fights as he would surely lose something in such a fight.

2.2 Purposes of a Move

Like in any game, also in go a good move has a purpose. In fact, good moves tend to have several purposes. Such purposes include: to attack an opponent's

group, to defend one's own group, to extend one's own *framework* (potential territory), to reduce the opponent's framework, to keep *sente* (initiative), to make a good *shape* (the formation of stones is efficient), to induce a bad shape for the opponent, etc. For example, "a sente move, which attacks an opponent's group in order to secure some territory while defending a weak group" sounds like a good efficient move.

These kinds of purposes represent the local aspect of the game. However, it is the global aspect of go which makes the game both complicated and intriguing. These global goals could include: to have at most one weak group at a time, to stay ahead in territory, to prevent the opponent's "sente moves, which attack my group(s) in order to secure some territory while defending his weak group", etc.

2.3 Making Efficient Moves

After the purposes for a move have been selected, it is necessary to find the best local move to accomplish the purposes–or if it seems that no move works, modifying or changing the purposes should take place. For example, saving a group might be the most important thing for the result of the game, but if there is no way to accomplish the feat currently, something else must be done.

It is quite common for stronger players to omit playing in some relatively important area if they do not find a good efficient move in the area or if they have difficulty deciding between seemingly equally good moves–they are delaying the decision on the local move further, until the game situation has changed so that the decision is easier.

2.4 Timing

The common feeling among weaker players is that the stronger player often seems to play the move the weaker player was planning to play next. And indeed, almost always the best time to play a move is the very last moment it can be played.

Figure 3 demonstrates the timing aspect. In the left diagram the starting position is shown with some possible moves for black to solidify his corner. We will have to assume that white is strong in both sides (outside the diagram). The middle diagram shows a very nice result for white, as he has managed to reduce blacks framework (moves 5 and 7) as well as leave behind the potential of making a living group in the corner with A. The right diagram shows the same moves but in the wrong order, as black has become sufficiently strong outside to resist white 5. It is a hard question, for example, when it is the correct time to play probing moves like 1 or valuable moves like A in the middle diagram.

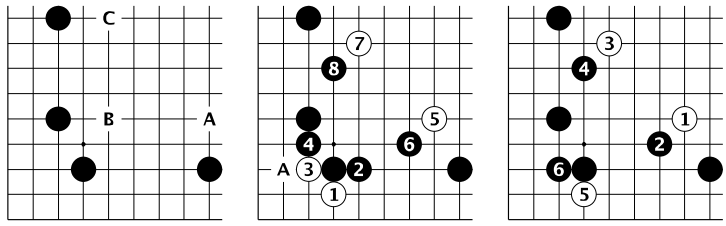

Fig. 3. A probe example.

3 Computer Go

Computer go has been studied nearly 40 years, with the current state of the art players attaining a strength around 7 kyu.[4] Here we will very briefly go through the different techniques used in constructing agents. For more thorough treatment, the reader is referred to [3,8,20], which cover the computer go field quite well (excluding the recent advances with Monte Carlo Go). A bibliography of go-related articles can be found in [11].

By looking at Figures 1 and 2 one can conclude that the gap between computers and the strongest humans is large. The reason for this gap is the unsuitability of the applied methods, which have been very successful with other games. We discuss evaluation functions in Section 3.1, move generators in Section 3.2 and divide-and-conquer approach in Section 3.3.

3.1 Evaluation Function

Using an evaluation function with a tree search has been a successful approach to many games–in go the difficulty of evaluating the game situation accurately and the vast number of possible moves have prevented a breakthrough. There are four main approaches.

The first and the traditional one is a manually crafted evaluation function, which combines the estimates of territories and influence.[5] Then simply the move leading to the best game situation is chosen. The thickness aspect is usually regarded by classifying moves to urgent and non-urgent moves, and always playing an urgent move if there is one.

[4] On KGS server (`http://www.goKGS.com/`) the best agents have stabilized around 4 kyu. In Finnish ranks this would most likely correspond to 6 kyu, but since there is not that much information on agents playing in real human tournaments and people tend to play less seriously on the internet, 7 kyu is probably a reasonable estimation.

[5] Influence is usually considered to represent, more or less, the probability of a given intersection belonging to a given player–which is not quite right. In practice the territory and influence estimation is implemented by letting stones radiate power to their surroundings and summing it up. With proper radiation functions and thresholds one gets numerical estimates for both territory and influence.

Another approach is to create (manually or automatically) a rule database, which directly gives a move without any game situation evaluation [22,24]–the advantage is the quickness and multitude of thumb rules (*proverbs*) around, which unfortunately hold only for the most of the time.

A black-box approach involves a training of a neural network to predict the winner for any game situation (for instance, see [7, 17, 23]). Neural networks do learn, but the common raw presentation of the game board along with some features has proven to be insufficient to construct a strong agent. NEUROGO [10] with its more complex board representation is the only pure neural network implementation with a moderate success.

The final approach is called Monte Carlo Go (MCG), in which simulations–i.e., random games[6]–are used to determine the move giving the best winning rate. Using an UCT algorithm [15] (running more simulations on the moves looking promising instead of running equal number of simulations for each move) and giving a proper bias for probable good moves have resulted in a very strong 9×9 agent [12]. The MCG/UCT approach is currently studied very actively in order to see, how well the approach will scale to 19×19 and how far the point of the diminishing returns is [31].[7] Based on their own experiments, the practitioners of MCG/UCT seem quite optimistic on both subjects.

3.2 Move Generators

Many methods for move generators have been already mentioned in previous sections, so there is not that much to be added here. The methods include rule-based move generators [22], neural networks [26] or goal/purposed-driven generators [34]. Usually each move generator gives a value for a move. The total value of a move on a certain intersection is a combination of the values given by different move generators.

The timing aspect of the game is not really modeled, although thinking in terms of temperature seems quite natural in this context. However, the published results (excluding the late part of the game) on the subject are scarce [6]. The temperature can be viewed as the point difference depending on which player makes a move to the area first. A point would here refer to a combined value of territory, influence and thickness resulting from the move–which makes the estimation of temperature difficult (in the endgame influence and thickness play no real role, so the estimation can be done accurately).

3.3 Divide-and-Conquer Approaches

A game of go has three phases: the opening (the board is divided into players' areas), the middle game (the areas are reduced, enlarged and exchanged) and the

[6] The game is played to the end using random moves (the legal moves are allowed, although some obviously bad moves are usually excluded).

[7] For discussions on these subjects the reader may consult the computer-go mailing list archives, `http://computer-go.org/pipermail/computer-go/`.

endgame (the exact boundaries of the areas are determined). Moderate success has been attained in opening [4,24], which is the most important part of the game–although, with amateur players the player who made the last big mistake tends to lose. The late endgame (where the board can be divided into independent subgames) can be considered to be solved [2] with combinatorial game theory [1]. The middle game has received the least attention and it is exactly the phase where the current agents are being outplayed by humans.

Besides these phases, go has some subgames as well, which are called problems. In life and death problems one should be able to determine the status of a group (alive, dead, depends on ko or whose turn it is) and find the correct moves to save or kill the group. In capture problems one tries to determine, whether given stones can be captured or not. And in connection problems one seeks for an answer, if a pair of stones/blocks/groups can be connected/disconnected. Several different approaches have been tried with some success, e.g., heuristics, databases, neural networks [9,16,28] and exhaustive search [14,29,30].

4 Playing Style

The current agents do not modify their way of play in accordance with their opponents or their experience. The common argument for this is that it is better to improve directly the agent against all the opponents (whereas learning has not been implemented because the current designs are not really suited for it). The argument is valid, but in our opinion a design which is capable of adjusting itself is a faster and more effective way towards a strong agent.

One could say the set of players' global goals and how he values them represent his style of play. Being strong at the local aspects of the game, tactics, is essential for a good player. But without striving towards a good style the player will not progress. Existence of styles manifests itself in the intransitive nature of the game, e.g., players A, B and C with similar strength might consistently win each other crosswise (A > B, B > C, but A < C).

We say a player's *playing style* is a set of global features and weighted hypotheses over these features. The global features themselves are relatively simple, like "number of opponent's weak groups", "agent's own estimated territory", "number of opponent's big ko threats", etc. For instance a hypothesis "number of my weak groups < number of the opponent's weak groups" could be part of a good style.

5 Tactician–Strategist Design

The nature of go is twofold: the local (tactical) battles and the global (strategic) war. If the global aspect is not handled properly, the war may be lost even if most of the local battles are won–and similarly, the war cannot be won without some prowess in the battles. Therefore we propose a two-part design consisting of a tactician and a strategist. The tactician's task is to determine the best local moves in every area (with respect to the current game situation) and their

expected results as a list of accomplished purposes. The strategist then chooses the most appropriate move (with respect to the strategist's playing style) from the ones proposed by the tactician.

5.1 Tactician

Section 3.2 lists some move generators that have been and are used for finding good moves. The tactician is quite similar to a set of move generators, but the relevant difference is how the purposes of different moves are combined. The tactician constructs a subgame consisting of a part of the board and the winning condition for the subgame is a list of purposes–if and only if every purpose is fulfilled, then the subgame is won.

As there are a huge number of possible subgames, we will assume that go is an incremental game most of the time. By incremental we mean that the result of a subgame rarely chances by moves played outside the subgame, hence after each move there are not that many subgames which have to be re-evaluated. However, every now and then the result of a subgame does depend on the rest of the board. These situations are studied in [25].

The tactician has failed, if it has not proposed a move in correct area or it has estimated the result of a subgame incorrectly. High-quality game records can be used to check, if the tactician proposes a move in the correct area–if not, either the actual move in the record is bad/irrelevant or the purpose of the move is unknown to the agent. The estimation errors in the subgame results can be detected in retrospect.

For implementing the tactician we would propose to use the move generators (see Section 3.2) to provide a list of moves along with their purposes. Nearby moves are grouped together into a same subgame and the move purposes become the winning condition for that subgame. This list of purposes is pruned until the subgame can be won (or there are no purposes left). For the estimation of the subgames' result we propose the MCG/UCT approach, since it is quite suitable for binary cases (the strongest play seems to result from maximizing the winning probability instead of maximizing the winning margin), i.e., we only want to know if all the listed purposes can be accomplished or not.

5.2 Strategist

There has been some research (besides the numerous studies how to improve the evaluation function) about how one should model the global part of the game [18,19,27]. The lack of success with these models may result from insufficient testing or failure to model the game well. Our proposal is to incorporate playing style into the agent (see Section 4) in a form of a strategist.

The strategist should receive a list of areas and purposes, which can be fulfilled by playing in the corresponding area. The strategist then chooses the area, which gives the best result in accordance with the strategist's playing style. The weights of the style can be adjusted by training against high-quality game records: given that a proper area of play was suggested by the tactician and it

was not chosen by the strategist, weights can be changed towards such values in which the correct area would have been selected. And with correct predictions the current weights can be reinforced. The hypotheses may be manually crafted, or the agent can try new random hypotheses out every now and then.

6 Conclusion

Like in any system design process, one must know what to aim at. Obviously, we would like to construct a strong agent, but it is useful if the agent can provide accessible reasons for its moves–for the benefit of detecting deficiencies and teaching beginners. In Sections 2 and 3 we provided some background to go and computer go, while pointing out that currently the biggest bottleneck is the middle game. We proposed a design aimed at the middle game. The advantage of the design is that self-learning can be incorporated into the system. Most of the current approaches suffer from their design, in which one may expect improvements only with manual tuning or by increased computing power. In addition, several different playing styles can co-exist, making it possible to choose a style according to the opponent.

On the other hand, the proposed design needs a good representation of the board–which is a difficult problem by itself (some discussion about the subject is given in [13, 21]). Also, managing the timing aspect may prove to be very challenging, i.e., how one can include temperature into the hypotheses in an effective way. And finally, how the subgames whose results depend on some other part of the board can be handled correctly.

Acknowledgements

The author thanks the Tekniikan edistämissäätiö and the Nokia Foundation for financial support. The author is grateful to his colleagues, especially Jouni Karvo, and anonymous referees for suggestions and corrections.

References

1. E.R. Berlekamp, J.H. Conway, and R.K. Guy, *Winning Ways for You Mathematical Plays*, Academic Press, London, 1982.
2. E.R. Berlekamp, and D. Wolfe, *Mathematical Go: Chilling Gets the Last Point*, A K Peters Ltd, Natick, 1994.
3. B. Bouzy, and T. Cazenave, Computer go: an AI-oriented survey, *Artificial Intelligence Journal* **132** (2001), 39–103.
4. B. Bouzy, and G. Chaslot, Bayesian generation and integration of k-nearest-neighbor patterns for 19x19 go, In *IEEE 2005 Symposium on Computational Intelligence in Games*, G. Kendall and Simon Lucas (Eds.), Colchester, 2005, pp. 176–181.
5. R. Bozulich, *The go player's almanac 2001*, Kiseido Publishing Company, Tokyo, 2001.

6. T. Cazenave, Comparative evaluation of strategies based on the values of direct threats, In *Board Games in Academia V*, Barcelona, 2002.
7. H.W. Chan, Application of temporal difference learning and supervised learning in the game of Go, Master's thesis, Chinese University of Hong Kong, 1996.
8. K. Chen, Computer Go: Knowledge, search, and move decision, *ICGA* **24** (2001), 203–215.
9. F.A. Dahl, Honte, a Go-playing program using neural nets, 1999.
10. M. Enzenberger, Evaluation in go by a neural network using soft segmentation, In *10th Advances in Computer Games conference*, 2003, pp. 97–108.
11. M. Enzenberger, Computer Go Bibliography, [23.5.2007], `http://www.cs.ualberta.ca/~emarkus/compgo_biblio/`
12. S. Gelly, Y. Wang, R. Munos, and O. Teytaud, Modifications of UCT with patterns in Monte-Carlo Go, Tech. Report 6062, INRIA, France, 2006.
13. T. Graepel, M. Goutrié, M. Krüger, and R. Herbrich, Learning on graphs in the game of go, *Lecture Notes in Computer Science* **2130** (2001), 347–352.
14. A. Kishimoto and M. Müller, Search versus knowledge for solving life and death problems in go, In *Twentieth National Conference on Artificial Intelligence (AAAI-05)*, 2005, pp. 1374–1379.
15. L. Kocsis, and C. Szepesvári, Bandit Based Monte-Carlo Planning, In *Proceedings of the 17th European Conference on Machine Learning*, Springer-Verlag, Berlin, LNCS/LNAI 4212, 2006, pp. 282–293.
16. B. Lee, Life-and-death problem solver in go, Technical Report CITR-TR-145, University of Auckland, 2004.
17. A. Lubberts, and R. Miikkulainen, Co-Evolving a Go-Playing Neural Network, In *2001 Genetic and Evolutionary Computation Conference Workshop Program (GECCO-2001)*, San Francisco, CA: Kaufmann, 2001, pp. 14–19.
18. A.B. Meijer, and H. Koppelaar, Pursuing abstract goals in the game of Go, In *13th Belgium-Netherlands Conference on Artificial Intelligence* (BNAIC 2001), Amsterdam, 2001.
19. A.B. Meijer, and H. Koppelaar, Towards multi-objective game theory – with application to go, in *4th International Conference on Intelligent Games and Simulation*, London, EUROSIS, 2003, pp. 243–250.
20. M. Müller, Computer go, *Artificial Intelligence* **134** (2002), 145–179.
21. L. Ralaivola, L. Wu, and P. Baldi, SVM and pattern-enriched common fate graphs for the game of go, in *ESANN 2005*, Bruges, 2005, pp. 27–29.
22. S. Sei, Memory-based approach in go-program KATSUNARI, In *Complex Games Lab Workshop*, I. Frank, H. Matsubara, M. Tajima, A. Yoshikawa, R. Grimbergen, and M. Müller (Eds.), Electrotechnical Laboratory, Machine Inference Group, Tsukuba, Japan, 1998.
23. K.O. Stanley, and R. Miikkulainen, Evolving a roving eye for Go, In *Genetic and Evolutionary Computation - GECCO 2004*, New York, Springer-Verlag, 2004, pp. 1226–1238.
24. D. Stern, R. Herbrich, and T. Graepel, Bayesian pattern ranking for move prediction in the game of go, In *ICML '06: Proceedings of the 23rd international conference on Machine learning*, ACM Press, New York, 2006, 873–880.
25. T. Thomsen, Lambda-search in game trees — with application to go, *Lecture Notes in Computer Science* **2063** (2001), 19–38.
26. E.C.D. van der Werf, J.W.H.M. Uiterwijk, E.O. Postma, and H.J. van den Herik, Local move prediction in Go, In *3rd International Conference on Computers and Games*, Edmonton, 2002.

27. S. Willmott, A. Bundy, J. Levine, and J. Richardson, Applying adversarial planning techniques to Go, *Theoretical Computer Science* **252** (2001), 45–82.
28. M.H.M. Winands, E.C.D. van der Werf, H.J. van den Herik, and J.W.H.M. Uiterwijk, Learning to predict life and death from go game records, In *Proceedings of JCIS 2003 7th Joint Conference on Information Sciences*, Ken Chen et al. (Eds.), Research Triangle Park, North Carolina, 2003, pp. 501–504.
29. T. Wolf, The program GoTools and its computer-generated tsume go database, In *The Game Programming Workshop in Japan '94*, M. Matsubara (Ed.), Hakone, 1994, pp. 84–96.
30. T. Wolf, Forward pruning and other heuristic search techniques in tsume go, *Information Sciences* **122** (2000), 59–76.
31. H. Yoshimoto, K. Yoshizoe, T. Kaneko, A. Kishimoto, and K. Taura, Monte Carlo has a way to go, In *Twenty-First National Conference on Artificial Intelligence (AAAI-06)*, Menlo Park, 2006.
32. R. van Zeist, *The Magic of Go* column 157, [23.5.2007], `http://shinbo.free.fr/TheMagicOfGo/index.php?tmog=157`
33. EGF official ratings, [13.12.2006], `http://gemma.ujf.cas.cz/~cieply/GO/gor.html`
34. GNU Go, [23.5.2007], `http://www.gnu.org/software/gnugo/`