

# Towards Incorporating Intent Inference into the Game of Go

Timothy T. Huang

Department of Mathematics and Computer Science  
Middlebury College  
Middlebury, VT 05753  
huang@middlebury.edu

## Abstract

Computer programs for the game of Go play only at the level of an advanced beginning player. The standard approach to constructing a program based on brute force game-tree search does not work well because of the game tree size and, more significantly, the difficulty in constructing fast, accurate heuristic evaluation functions. In this paper, we consider the use of intent inference in a Go program. In particular, we discuss how models of an opponent's long-term playing style and short-term intentions can direct the exploration of candidate moves and influence the evaluation of game positions. We propose a probabilistic approach to user modeling and intent inference, and we note key issues relevant to the implementation of an intent inference agent.

## Introduction

The ancient game of Go is one of the most widely played strategy games in the world, especially in the Far East. More effort has been spent on developing computer programs that play Go than on developing programs for any other strategy game except chess. Unlike chess, however, where the best programs play as well as or better than the top human players, the best Go programs play only at the level of advanced beginners. Standard game-playing techniques based on brute force minimax search are not directly useful for Go because accurate evaluation functions are slow and because the game tree is extremely large. Hence, most current programs rely on very limited use of search and on carefully tuned heuristics for choosing a subset of legal moves to fully evaluate.

We believe that the performance of current Go programs can be improved by having them reason explicitly about an opponent's playing style and intentions. This idea is supported by the fact that professional players will comment on their opponents' intentions when writing post-game commentary, and by the fact that the IBM chess computer Deep Blue's successful defeat of former world champion Garry Kasparov involved specific tuning for Kasparov's playing style.

We begin this paper by presenting an overview of game play in Go and the factors that make computer Go challenging. After motivating the use of intent inference in Go, we consider the components of an intent inference agent: the inputs it can be given, the possible outputs it can produce, and the agent architecture that might generate those outputs. We discuss how an architecture based on dynamic probabilistic networks seems well-suited for intent inference, and we describe how inferred intentions can be used to influence move selection. Finally, we compare intent inference in Go with intent inference in other fields.

## The Game of Go

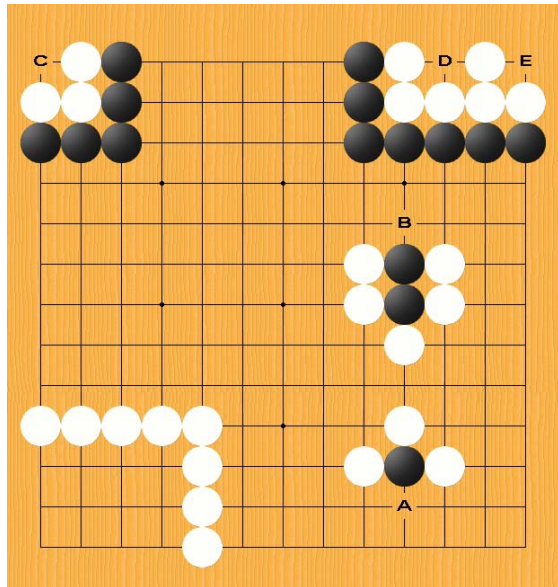
To start, we present a brief description of how Go is played. A comprehensive explanation of the rules can be found in (Iwamoto 1972) or (Kim and Soo-hyun 1997); our aim is primarily to give a sense of the goal of the game and the factors that make it both enjoyable to play and difficult to master. We also discuss the characteristics of Go that make it such a perplexing computational problem.

## Playing Go

Go is played with black and white stones on a board of  $19 \times 19$  intersecting lines (for variation, smaller grids such as  $9 \times 9$  or  $13 \times 13$  are sometimes used). Starting with an empty board, two players take turns placing one stone at a time onto one of the 361 intersections. The goal is to acquire territory (empty intersections) by surrounding it with stones. Once placed, stones do not move; however, a player can capture stones of the opposing color by surrounding them completely on all four sides. The game ends when both players pass. The player who has surrounded the most points of territory wins the game. In tournament play, there are no draws because the white player receives 5.5 additional points (called *komi*) to compensate for playing second.

Consider the stones shown in Figure 1. In the bottom left corner, the white stones have surrounded 12 points (intersections) of territory. Towards the bottom right, the black stone is surrounded on three of four sides by white stones. White can capture this stone by playing at the intersection marked **A**. Above those stones, two black

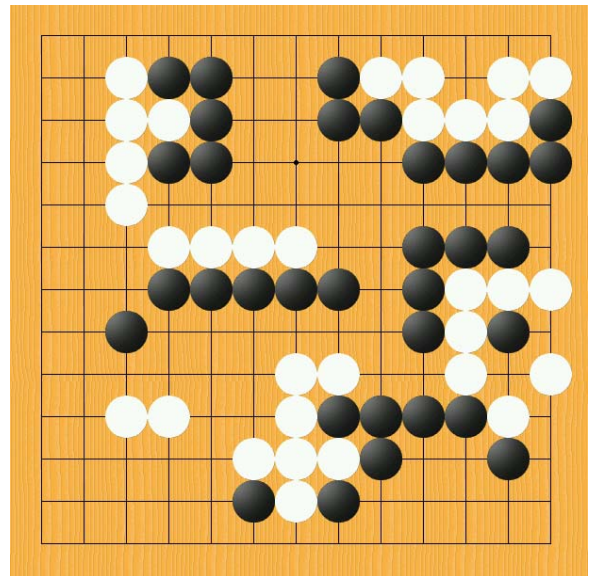
stones are almost fully surrounded by white stones. White can capture them by playing at **B**. If it is black's turn, black can help those stones escape by playing at **B** first. In the upper left corner, black can capture the white stones by playing at **C**, thereby fully surrounding the three stones. In the upper right corner, the white stones are *alive*, i.e., safe from capture, because black cannot play a stone at **D** and a stone at **E** simultaneously.



**Figure 1: White can capture one black stone by playing at A or two stones by playing at B. Black can capture three white stones by playing at C. Since black cannot play at D and E simultaneously, the white stones in the upper right corner are alive.**

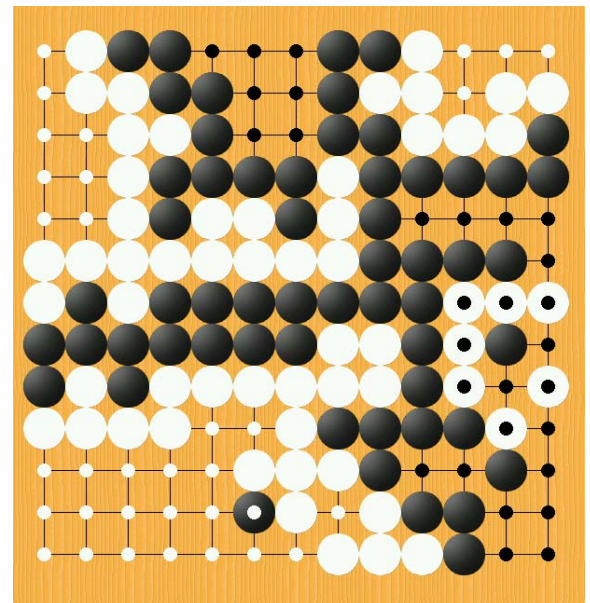
The stones in Figure 1 were artificially arranged for explanatory purposes. Figure 2 shows a more realistic board situation partway through a game on a 13 x 13 board. In this game, white has staked out territory in the upper right, upper left, and lower left sections of the board. Meanwhile, black has staked out territory in the lower right and upper middle sections of the board and has pushed into the middle left section (thus reducing white's territorial gain). Furthermore, black has effectively captured the white stones in the middle right section of the board (see (Kim and Soo-hyun 1997) for a full explanation).

Part of what makes Go so engrossing and challenging is the interplay between tactics and strategy. On one hand, players fight localized battles for territory in small regions of the board. On the other hand, the arrangement of stones on one side of the board affects the strength and usefulness of stones on the other side of the board.



**Figure 2: White's turn to move partway through a closely-played example game on a 13 x 13 board.**

Figure 3 shows the board situation at the end of the same game from Figure 2. Black's and white's territorial points are marked with small black and white dots, respectively. The final score, taking into account captured stones and komi, has white ahead by 4.5 points of territory.



**Figure 3: The final board position. Taking into account captured stones and komi, white wins by 4.5 points.**

### Computer Go

From a computer scientist's perspective, what stands out about Go is that it remains a perplexing computational

problem. The best computer programs for Connect-4, Mancala, Othello, checkers, chess, backgammon, and Scrabble all play at or better than the level of the best human players. Although more time and effort has been spent developing computer programs for Go than for any of these other games except chess, any reasonably skilled club player can defeat the best computer Go programs (Müller 2000).

Most programs for strategy games, including those for chess and checkers, utilize some variation of brute-force game-tree search: the programs examine all sequences of play up to a certain number of moves and then choose the move that seems to lead to the most advantageous position. This approach does not work well with Go for two reasons. First, the search tree for Go is exponentially larger than other games. For example, the average branching factor (number of available legal moves) in chess is about 35, whereas the average branching factor in Go is about 250. Second, and more significantly, competent heuristic evaluation functions for Go are far more computationally expensive than for other games. Unlike chess, where the difference in material is easily computed and provides a reasonable estimate of which side is winning, Go positions have no easily extracted features from which to quickly estimate the game state.

Researchers throughout the world continue to work on computer Go and gather regularly at computer Go tournaments to pit their updated programs against each other (Hafner 2002). While they continue to make incremental improvements, many believe that a significant breakthrough, perhaps one involving much better recognition of good and bad patterns of stones, is necessary for substantial progress.

## Motivation

It may not be immediately clear that a computer Go program would benefit from reasoning explicitly about an opponent's intentions. After all, any program already has full access to the complete state of the game, and both players share the same, well-defined objective, namely, to acquire more territory than the other player. Viewed as a brute force minimax search problem, the task of choosing the best move seems to leave no room for intent inference.

Several factors, however, suggest that user modeling and intent inference can be beneficial. First, and perhaps most importantly, choosing what move to make involves directing the use of limited time and computational resources. Although unnecessary if it were possible for a Go program to fully expand the game tree and perform minimax backup of the game-theoretic values at the leaves, intent inference may be very valuable in directing a selective search of the game tree or estimating the value of a particular node.

Second, user modeling has been utilized before with much apparent benefit. For the 1997 chess match between IBM's Deep Blue chess computer and then-world champion Garry Kasparov, the IBM programming team

worked with chess Grandmaster Joel Benjamin to study Kasparov's past games and tune Deep Blue specifically for its match with him. They even tweaked the program in between games, leading some to suggest that the team was cheating (Chelminski 2001).

Third, professional Go players seem to utilize user modeling and intent inference in their own play and in analyzing the play of others. For example, the Japanese professional O Rissei, commenting on a March 2000 game he played as white against Cho Chikun, notes at one point, "Black could play 37 at A, but Cho is not one to approach the corner from that direction. If he approached at all, it would be from the other side at B, to attack the white group above."<sup>1</sup> Clearly, O Rissei's knowledge of Cho Chikun's playing style influenced his thinking. Commenting on a June 2002 game at the World Amateur Go Championship, Yuan Zhou notes, "White 16 shows White's attitude toward Black 15, that is, *White welcomes a fight* around here... *Black has decided to take the bottom territory* with Black 17... Black 55: *Black wants the game to become a territory race...*" (italics mine). Zhou describes each player's plans and intentions to help explain their current and subsequent moves.

## An Intent Inference Agent

Having considered reasons why intent inference holds promise for improving the performance of computer Go programs, we now examine the various aspects of an intent inference agent. In particular, we describe what inputs will be available to the agent and what outputs such an agent might produce. We also consider the architecture for mapping from inputs to outputs, and we propose a model architecture based on dynamic probabilistic networks.

### Input

The input is fairly straightforward. Over the course of a game, it consists of each move played. It might also include how much time was spent on each move, how much time remains on each player's clock, how many pieces each side has captured, and an estimate of the current score. Information concerning a human opponent's body language might actually be very valuable for intent inference, but this is beyond the reach of current biometric technology. For generating a longer-term user profile of a particular opponent's playing style, the agent would also have access to records of the opponent's past games.

The inputs are really no different from the inputs required by any computer Go program. By explicitly considering an intent inference component, however, we acknowledge the possible benefit of higher-level features about an opponent's intentions, and we recognize the usefulness of building and updating long-term models of opponent playing styles.

---

<sup>1</sup> Here, 37 refers to the 37<sup>th</sup> move of the game, while A and B refer to locations on an annotated board diagram (not shown in this paper).

## Output

The aim of the intent inference agent is to provide descriptive output that can improve the Go program's decision-making ability. Many reasonable alternatives for this output exist, so in this subsection we classify some of the possibilities and provide specific examples.

If a player's historical game records are available, some output may pertain to an opponent's playing style or long-term behavior. This output may consist of high-level symbolic descriptions. For example, some players may be characterized as "territory-oriented"; they tend to quickly grab and defend territory along the corners and sides of the board. Other players may be described as "influence-oriented"; they prefer to build strong stone formations that themselves do not acquire territory but that can be used to attack powerfully, eventually securing territory. Furthermore, some players enjoy highly tactical, fighting situations, while others prefer quieter, more strategic play. Other output may provide specific, low-level descriptions by inferring, for example, that a particular player often starts with the "Chinese Opening" or prefers corner sequences of play that involve playing at the "3-3 point".

By contrast, and perhaps more frequently the case for intent inference, some output may describe an opponent's intentions for just the next move, the next several moves, or for the current game. This output can also involve high-level or low-level descriptions. On a high level, the agent might infer that the opponent welcomes an attack in the lower portion of the board, or that the opponent is trying to pretend that he wants the upper right section of the board, when in fact he wants the upper left section, or that the opponent is playing especially aggressively or defensively. On a lower level, the agent might infer that the opponent is likely to play at the "3-4 point" in the lower left corner on this turn or shortly thereafter.

## Agent Model

Among the myriad agent architectures that could be used for user modeling and intent inference in Go – expert systems, neural networks, and decision trees, to name a few – we believe that a probabilistic model provides the most appropriate architecture. We propose this model not to address uncertainty in our knowledge of the game state or to handle the complexity of the environment, but to model our uncertainty of a particular opponent's intentions and to predict what type of move that opponent will choose, e.g., attacking vs. defending, responding to a threat vs. ignoring it and playing elsewhere, etc.

The particular model we propose is the dynamic probabilistic network (DPN), a representation scheme that factors the state of an environment into a finite set of state variables, explicitly represents conditional dependence relationships among the state variables using a directed acyclic graph, and models their evolution over time (Dean and Kanazawa 1988). When specific values are observed

for some of the variables in a DPN, posterior probability distributions can be computed efficiently for any of the others using a variety of inference algorithms (Pearl 1988).

A DPN model is well-suited to intent inference in Go for several reasons: First, the short-term intentions and long-term style of an opponent will change over the course of a game and from game to game, respectively, and a DPN explicitly models this stochastic process. Second, the network structure and associated probability tables provide a natural way to incorporate high-level Go knowledge and to interpret the inference algorithm results. Third, an accurate, compact approximation of the belief state can be maintained even over multiple moves and multiple games (Boyer and Koller 1998).

Numerous issues must be addressed in specifying the actual DPN structure for intent inference. Will the possible values for observable variables be specific intersections on the Go board, or will they be more general descriptions of a move's location? Should there be other observable variables corresponding to higher-level features, such as a played move's relationship to other stones in the same area, or should the actual pieces on the board be included in the current state? Should the opponent's intention be represented as a single state, or can it be factored into smaller components? We recognize that substantial work remains and are currently exploring these questions.

## Using the Output

There are numerous possible ways in which information about an opponent's long-term playing style and short-term intentions can be utilized in choosing a move, and we sketch two of them here.

First, the information can be used to direct computational effort toward evaluating the state of likely future positions and considering appropriate moves to counter an opponent's plans. As a simple application, suppose a Go program performs highly selective search as part of its move choice process. Before playing a particular opponent, it can examine the historical games of that opponent and categorize past moves into two or more pre-determined classes, e.g., "attacking" vs. "defending". The frequency with which moves fall into each category provides a rough picture of that opponent's playing style. During a game, it can direct the expansion of the game tree by categorizing possible opponent moves and expanding branches corresponding to moves that more closely match the opponent's playing style.

This approach also applies to short-term, inferred intentions. Suppose that over the course of a single game, the DPN-based agent infers that the opponent is trying to capture a group of stones or establish a strong base in a particular region of the board. The search could be directed toward more extensive consideration of responses to those efforts. Alternately, if a program judges a game in progress to be almost even but infers that its opponent thinks he or she is winning and intends to seal the win by playing conservatively, it might give more consideration to moves

through which the opponent could lose points by responding too cautiously.

Second, information about inferred intentions can be used as parameters to a heuristic evaluation function on non-terminal board positions. In general, it does not make sense to assign different evaluations to the same board configuration simply due to different presumed intentions. However, current Go evaluation functions often misjudge the significance of unsettled stone patterns in the middle game, and reasonable inferences about an opponent's intentions for playing near such patterns may improve an evaluation function's accuracy by biasing its assessment of the status of those stones.

### Relationship to Other Fields

Not surprisingly, Go as a problem domain for intent inference differs substantially from other domains. In Go, there are only two individual opponents, one player on each side; there is no uncertainty about the state of the world, which is fully observable to both sides; and, both players have the same objective and know exactly what the other player's objective is. By contrast, in a military situation, for example, both sides typically involve teams of agents, each of which may have differing individual intentions; there may be substantial uncertainty about the state of the world due to noisy and incomplete sensor information; and, the other side's objectives, both as a team and as individuals, may not be clear and may change over time (Bell, Santos, and Brown 2002).

At the same time, Go shares certain similarities with the military domain. Both concern adversarial situations, both involve acquiring and holding onto territory, both have elements of deception, both can involve probing and sacrificial actions, and both may involve changing short-term objectives.

More importantly, work on intent inference in Go may provide useful theoretical and experimental results about approaches that may be applicable to other fields. A successful DPN model for intent inference in Go may shed light on how the DPN architecture can be effective for other intent inference tasks that can be modeled as stochastic dynamic processes, and techniques developed for guiding search and influencing heuristic evaluation functions may have broader impact as well. Effective applications of intent inference to Go may even provide insight on how intent inference can best be useful in military strategy.

### Summary and Future Work

In this paper, we described the game of Go and argued that intent inference may help improve the performance of computer Go programs. We discussed the components of an intent inference agent, namely, the inputs, the outputs, and the agent model, and we suggested that dynamic probabilistic networks provide an agent architecture that is

well-suited for intent inference in Go. Finally, we discussed ways in which the output of an intent inference agent could be brought to bear in choosing a move, and we explored connections with other fields.

Implementing a full Go-playing system is a task where the devil is in the details. Clearly, much work remains to be done. Moving from modeling to implementation will require further investigation and refinement of our ideas concerning the most informative forms of descriptive output, the various approaches for constructing the DPN structure and probability tables, and the ways in which inferred intentions can be utilized in the process of move selection.

### Acknowledgements

This work was supported by the National Science Foundation under Grant No. 9876181, and by Middlebury College.

### References

- Bell, B. and Santos Jr., E. and Brown, S. 2002. Making Adversary Decision Modeling Tractable with Intent Inference and Information Fusion. In *Proceedings of the 11<sup>th</sup> Conference on Computer Generated Forces and Behavioral Representation*.
- Boyen, X. and Koller, D. 1998. Tractable Inference for Complex Stochastic Processes. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*: 33-42.
- Chelminski, R. 2001. This Time It's Personal. In *Wired* 9.10.
- Dean, T. and Kanazawa, K. 1988. Probabilistic Temporal Reasoning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*: 524-528.
- Hafner, K. 2002. In an Ancient Game, Computing's Future. In *New York Times Online*. Internet. Available WWW: <http://www.nytimes.com/2002/08/01/technology/circuits/01GON E.html>.
- Iwamoto, K. 1972. *Go for Beginners*. Ishi Press.
- Kim, J. and Soo-hyun, J. 1997. *Learn to Play Go, Volume 1, 2<sup>nd</sup> Edition*. Good Move Press.
- Müller, M. 2000. Computer Go. In *Artificial Intelligence Journal* 134(1-2): 145-179.
- Pearl, J. 1988. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann.
- Rissei, O. 2000. The 24<sup>th</sup> Kisei Title Match: Game Six. In *Go World* 89:32-36.
- Zhou, Y. 2002. The Decisive Game in the 2002 World Amateur Go Championship. In *American Go Journal* 36(3): 26-28.