A LEARNING ARCHITECTURE FOR THE GAME OF GO

A.B. Meijer and H. Koppelaar
Delft University of Technology. Faculty ITS. Section Mediamatics.
Mekelweg 4, P.O.Box 356, 2600 AJ, Delft, The Netherlands.
{a.b.meijer, h.koppelaar}@its.tudelft.nl

ABSTRACT

In this paper, a three-component architecture of a learning environment for *Go* is sketched, which can be applied to any two-player, deterministic, full information, partizan, combinatorial game. The architecture called HUGO has natural and human-like reasoning components. Its most abstract component deals with the selection of subgames of *Go*. The second component is concerned with initiative. The notion of *gote no sente* (a move that loses initiative but creates new lines of play that will hold initiative) is formalized. In the third component, game values are computed with a new kind of α-β algorithm based on fuzzy, partial ordering. Our approach leaves some valuable control parameters and offers ways to apply further machine learning techniques.

KEYWORDS Combinatorial Games, Uncertainty, Initiative, Fuzzy Partial Ordering, Game of Go

INTRODUCTION

Two-player, deterministic, complete, information, partizan, combinatorial games form a family of games which has received a lot of attention from the AI community over the last decennia. Altough much progress has been made, resulting in some well-playing chess programs for example, almost all modern programs for these games lack (well-formalized) human-like behavior in general and the notion of initiative in particular. This need not result in a poor performance, rather it depends on the domain. We hypothesize that the notion of initiative is mandatory for writing a good program for the most notorious of combinatorial games: *Go*. It's folklore is full of terminology concerning initiative and professionals play the game at an abstract level of initiative, far beyond considering just move sequences, as in the commonly used α-β algorithm (or any other minimax-based search algorithm). Furthermore, the α-β algorithm is based on some absolute scalar-valued evaluation function, whereas humans can be said to use an ordering function to compare two board situations directly with one another. For example, a professional *Go* player could reason that one position is slightly better than another when his walls look a bit thicker (=stronger).

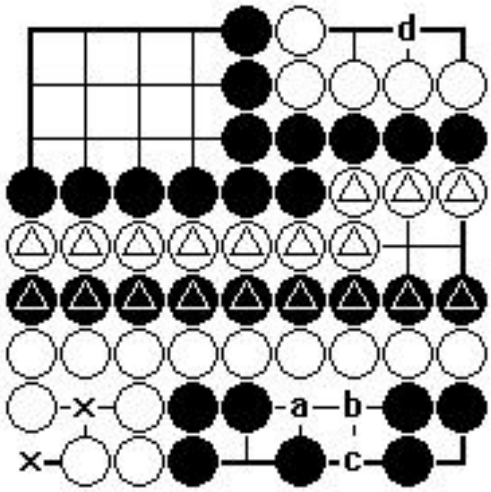

Figure 1: An endgame situation in 9×9 *Go*.

This paper is organized as follows. In the next two sections we will give a short introduction to the game of *Go* and combinatorial game theory. Then, we formalize these two human-like concepts that arise in the learning environment of *Go* and embed them in an architecture called HUGO. The following three sections further explain the three components of HUGO. The first component deals with the selection of subgames of *Go*, the second with initiative and the third with the computation of game values. We end with some concluding remarks and future work. Go terminology is written in

SMALL CAPS.

THE GAME OF *GO*

Go is an ancient game, originated in China about 4000 years ago. It has influenced oriental warfare, which shows off in the shape of the Great Walls of China and, very recently in Afghanistan, in the preference for semi-fixed frontlines between the opposing factions, which only change hands if there is a broad momentum in favour of one of the sides. Compare this with *Go*, where two players have to embark territory by alternately placing a stone on a grid, gradually building strongholds and eventually walls that completely surround one's territory. Strongholds will only be given up if the opponent has created enough influence (*Go* term for momentum) to walk over it.
The rules of *Go* are very simple in principle (but in finesse they can vary a lot over different rule sets like the Chinese, Japanese or mathematical *Go* rules). The *capturing rule* is the most important, stating that a string of stones gets captured if all of its neighboring intersections are occupied by enemy stones. This rule implies that the two x's in figure 1 are suicide, which implies in turn that the white group that surrounds them cannot be captured (*Go* terminology: the group LIVES). The white group can only be captured if White would cooperate foolishly and plays on one of the x's himself. Black is then allowed to play the "temporary suicide" of the other x, because this would capture the entire white group and the suicide is resolved.
The goal of the game is to make living groups that surround more territory than your opponent.
If in the game of figure 1 Black were to play, he would have two good options. The first is to play at d, killing the white group since it has become impossible for White to construct *two* suicide points (EYES) like the two x's. The second option is to save his own group in the bottom right by playing at a. This would result in two black strings, each having one eye. The strings can always be connected with White b, Black c or vice versa. The resulting group lives with two eyes. The best of the two is a, since there are more stones in this group.

COMBINATORIAL GAME THEORY

This section is a very short introduction to Combinatorial Game Theory, loosely following (Cazenave, 1996). It is a mathematical theory for games and numbers, developed by J.H. Conway (Conway, 1976) and adapted to many games by Berlekamp, Conway and Guy (Berlekamp et alii, 1982).

Definition 1 *A combinatorial game $G = \{F|O\}$ is composed of two sets F and O of combinatorial games. Every combinatorial game is constructed this way.*

In games, F should be seen as the set of options (board positions) that player *Friend* can reach with one legal move. O can be looked at as the options for player *Opponent*. F can have two possible values, W (win for Friend) or L (loss for Friend, so win for Opponent). If Friend has a legal move that ensures a win for the whole game, then the value of F is W. This gives four possible outcomes for a combinatorial game: WW, WL, LL and LW (We will use both WW and W|W as abbreviated notations for {W|W}). The left half of a game value is the maximum result that Friend can obtain, the right half is Opponent's best result.
WW denotes a game that is won by Friend, irrespective of who moves first (both player can at best move the game to W = win for Friend). A *Go* example is the white group in the bottom left of figure 1, which lives unconditionally.
WL is an unsettled game, it is won by the player who moves first. An example is the life status of the white group in the upper right corner. If White moves first he can play at d, resulting in a living shape (its territory contains two eyes, intersections that are suicide for Black).
LL is a lost game for Friend, so a sure win for Opponent, even if Friend moves first. Trying to kill the black group in the upper left corner is a lost game for White (given correct counterplay from Black)
LW is a somewhat strange equilibrium situation where the player who moves first will lose the game. An example of this situation is known as SEKI. The triangled stones in figure 1 form a SEKI: either player who wants to capture the opponent string of triangled stones and plays at one of the two shared intersections will immediately be captured himself.
To evaluate a subgame, we define

$$Value(W) = 1, \quad Value(L) = -1.$$

In order to evaluate the global position, each subgame is assigned a numerical importance. A (linear) combination of the value of the subgames yields an indication of the balance of power. The precise nature of combination is a task for component 1.
Now it is possible to define the value of a move:

$$Value\,(move) = Value\,(F) - Value\,(O)\,.$$

$Value(move)$ is the value of the move that achieves state F and thus prevents Opponent from achieving state O. The value of a SEKI needs special attention, since usually there will be no play inside a seki (the value for such a move is -2 times the numerical importance). Its value depends on the rules. In *Chinese* counting one counts one point for all the stones that live in a SEKI but no point for territory, which is natural since no player controls the in-between territory.
Another commonly used term for game value is temperature, which corresponds to the size of the largest play.

As the game progresses, the temperature tends to drop. *Local* temperature corresponds to the size of the largest play in a region of the board. *Ambient* temperature is the temperature of the game besides the local region.
In theory, the value of a game has only two values, W or L. However, it is often intractable to compute the precise game value. Cazenave therefore extended Conway's theory to uncertain outcomes (Cazenave 1996). He introduced a variable U, denoting an uncertain game value with range [-1, 1]. If halfway a game the position is considered roughly equal for both players, then it makes sense to assign this uncertain position the value 0.
U can be seen as a control parameter along the risky-safe axis, since one is free to define the value of U. A low value for U would result in conservative play, a high value models a form of risky play.

THE HUGO ARCHITECTURE

As stated in the introduction, it is our aim to model human concepts that exist in the folklore of *Go*. We have identified three components that are inadmissable in our approach. The first component is to select relevant subgames and their numerical importances. The second is an initiative engine and the third computes game values. We have called the resulting architecture HUGO (human *Go*), see figure 2.
HUGO's input is a board position and some constraints, for example a time or memory limit. This input goes to component 1, which outputs a collection of games. This is the input of the initiative engine, which calls the third component to calculate the value of every single game. Knowing all the game values, the initiative engine tries to find a move that both scores some points and holds initiative.

COMPONENT 1: CHOICE OF SUBGAMES

The task of this component is to select a collection of well-defined subgames (or simply: games) of *Go* which form the basis for further computation. A good choice of games should have a high discriminative ability: the player that wins most of the important games should also win the whole game and vice versa. The problem of such an approach is that the subgames are not independent and thus the value of the whole game is not simply the sum of the value of the subgames (Berlekamp et alii 1982). However, advanced human *Go* players constantly use a variety of games, such as life or death, connection of stones, territory and more. They do this as a means to obtain overview in a chaotic board situation and the result is some sort of a mental model of the state of the game. For each separate game, they can quite easily find the (sub-)optimal move. The difficult part is to take the interactions among games into consideration and to find the optimal move for the whole game. Playing *Go* at this abstract level means finding conflicting interactions all the time and then resolving or exploiting them.

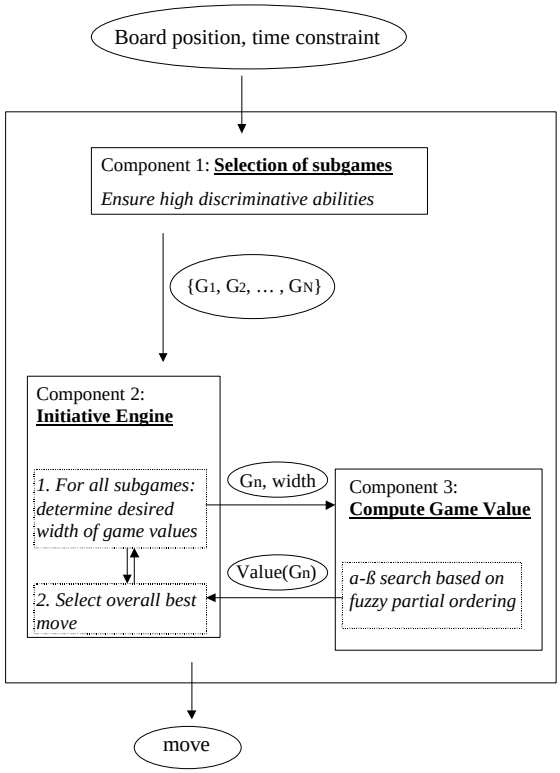

Figure 2: The HUGO architecture

A good collection of games to start with are (1) life and death for strings of stones and (2) connection of strings (the fundamental importance of these games follows from the importance of the capturing rule). These two families of games govern tactical play, but are also inadmissable in strategic play.
One can think of many more games and there is indeed a lot of work to be done, before this can all be automated. However, one could also leave this task to an expert. Introducing more families of games would enhance more sophisticated play, but at the cost of a higher computational effort.
One further sophistication is to define connections between groups (clusters of strings that are highly interconnected) instead of strings of stones. A clustering algorithm would be valuable for this task.

COMPONENT 2: INITIATIVE ENGINE

Given a collection of well-defined subgames from component 1, the task of the initiative engine is to find the move that yields the most points, preferably while hold-

ing initiative. When one has the initiative, one dictates the course of the game and one can choose to accumulate small gains from different subgames.

The first step of this component is to compute the value of each game. A sophisticated game value should not only reveal the player how will win a particular game with which move, but should also tell what are threatening moves in the game. A simple approach to meet this criterium is to compute the game value for the two cases that each player once moves first and once moves second (this is more or less common practice in the more sophisticated current *Go* programs). If one wishes to model threats, s/he must also compute game values for two cases where each player gets to play twice before the opponent may respond.

Let us call the number of plays by one player before the opponent answers the *width* of a game value. One possible outcome for a game value with width two is WL|LL. This is the situation that Friend can move to an unsettled situation (WL) which he can win if he gets the chance to play a second move (only when Opponent ignores the first move), whereas Opponent can move to a won game (LL) if he does answer. Cazenave already derived that this type of game has a threat for Friend only (Cazenave 1996). In fact, the general representation of a one-sided threat is W*UUU* for Friend and *UUU*L for Opponent. W*UU*L is a two-sided threat.

If Friend wishes to play KIKU (play to hold initiative) he can choose to play a series of threatening moves. Although Opponent simply can answer all the threats and win in all the threatened games, this line of play can be advantageous to Friend because a move generally plays in more than one game at the time (multipurpose moves). If a move is a threat in two games at the same time, it will be difficult (if possible at all) for Opponent to find a move that resolves both threats at the same time. The simplest line of follow up play is to answer the most important threat, but then Friend can win the other game.

Sometimes there is such a big move that one does not care about holding initiative but just grabs the points associated with the move (for example killing a large group). In general, this should be done when the local temperature is hotter than the ambient temperature. Such moves usually end in GOTE (loss of initiative) and the resulting game is cold (not much to gain left for either side). In general, GOTE play lowers the local temperature, while SENTE play raises it.

RYO-SENTE (two-sided SENTE) arises when a move is a two-sided threat. If it arises in important games it is usually played immediately.

Despite losing initiative, GOTE moves can develop new initiative. If a move is gote in one game but creates new sente moves in another game, then this move is called GOTE NO SENTE (GOTE with SENTE potential). For example, WLLL|LLLL is a lost game for Friend, but he can change the game to a (lost) game where he does have a threat (WLLL), whereas Opponent can move to a game which he has won and leaves no threat (LLLL). So, an example of a GOTE NO SENTE move is one that is GOTE in one game and moves another game from WLLL|LLLL to WLLL. GOTE NO SENTE moves can only be found if one considers the possible outcomes if a player moves three times in a row before his opponent starts to answer (width = 3). None of the current *Go* programs described in literature does this. Cazenave uses game values up to width 2 (but gave a formalization for greater width).

Computing up to width K is twice the cost of computing up to width *K-1*. Therefore, one has to know when to do so. This task should be dealt with by component 2. One good possibility is iteratively widening based on game value feedback from component 3, since it is known from (Berlekamp et alii, 1982) that one best makes moves in hot (unsettled) games. So these are the games whose value one would like to know most precise.

All the current *commercial* programs lack a well-defined notion of initiative and their advantage in playing strength can be expected to shrink relative to programs based on scientific research, which do implement Combinatorial Game Theory.

COMPONENT 3: COMPUTING GAME VALUES

The task of this component is to compute the game-theoretic value of a particular game. Once determined to what width an outcome has to be calculated, one actually has to calculate the value of the game and the move(s) that accomplishes this value. In fact, in order to do so, one has to calculate the value for all the moves and select the best one. However, at this point it hasn't yet been determined in which game is to be played. It is better not to remember just the best move but all (good) options, because this leaves the possibility open to detect multipurpose moves. Such a move might be suboptimal in all the games it plays in, but can be superior in the combined game compared to any of the optimal moves in the separate games. Furthermore, remembering all good options can offer a means to prevent unnecessary recalculations. Imagine you would only remember the best option (and also only the best reply) and your opponent plays in a later stadium a suboptimal move in the nearby region, threatening the result of your best move. You then would need to re-evaluate this move, whereas you wouldn't need to, had you remembered opponent's suboptimal moves too.

So the task in this component is to calculate the value of a game, given some width. An α-β algorithm (or other forms of minimax search) can be used to do this, but this requires some sort of scalar-valued evaluation function to compare two board positions. This indirect comparison has several drawbacks, including the horizon effect. Quiescence search has been invented to circumvent this

problem, but does not solve all the problems. This and other disadvantages of using absolute evaluation functions are well described in (Müller 2000).
Müller concludes that in many cases partial ordering is better than absolute evaluation. This makes sense, since it is more natural and even in α-β search absolute evaluation is used for partial ordering in the end.
We hypothesize that α-β search in its current form cannot handle the multidisciplinary and strategic nature of *Go*.
Müller's alternative was a method that combines partial order evaluation with minimax search, called *Partial Order Bounding*. In this method one categorizes all the possible states into a success set or a failure set. Minimax search is performed to determine which move guarantees that any leaf node of this move's subtree belongs to the success set. Leaf nodes are evaluated just by checking whether or not it falls into the success set, so the minimax values are boolean.
In contrast to this, our approach does not directly use a success set, rather we bring partial ordering right into the α-β algorithm itself, so we use no boolean-valued evaluation function. To be more precise, we will use fuzzy partial ordering of vector states. Obviously, this generalization is not possible without considering all consequences of introducing fuzzyness and partial ordering at the same time.
In short, fuzzyness can be dealt with by putting states that are approximately equal into one cluster. If one watches out that intra-cluster distances remain (far) bigger than inter-cluster distances, it is possible to order clusters just as if one were ordering single board situations.
The partial ordering function compares the game value of two (clusters of) feature vectors $\mathbf{a}$ and $\mathbf{b}$ and outputs an ordering such as $\mathbf{a} \succ \mathbf{b}$ ($\mathbf{a}$ is preferred over $\mathbf{b}$), $\mathbf{a} \succ\succ \mathbf{b}$ ($\mathbf{a}$ is by far preferred over $\mathbf{b}$) $\mathbf{a} \approx \mathbf{b}$ (about equal), $\mathbf{a} \sim \mathbf{b}$ ($\mathbf{a}$ and $\mathbf{b}$ are incomparable), combinations of these like $\mathbf{a} \succsim \mathbf{b}$ ($\mathbf{a}$ is preferred over or approximately equal to $\mathbf{b}$) and negations. Notice that incomparable is different from approximate equality. Two states that have approximately the same value are incomparable if one is far hotter (more unstable) than the other.
A traditional α-β algorithm (without speed enhancements) remembers during search just the values α and β. In comparison, our generalized algorithm has to remind a partially ordered tree of fuzzy clusters of already evaluated options. We believe that the considerable extra amount of work (firm theory plus implementation) is worthwhile, since it offers a natural (=human) look at *Go*. For example, it enables an ordering between two moves based on the achievements relative to some shared parent state, which is really different (and probably easier) than any scalar-valued board evaluation technique. The details of this approach will be discussed in a forthcoming paper, due to shortage of space here. One issue that will be discussed in great detail and with mathematical rigour is omitted here, namely the fact that game values are looked at as scalars in the above discussion, whereas they are more like an interval [O,F], where O is an underbound for the real value (the best result for Opponent) and F is an upperbound (Friend best result). The real value is determined as soon as one of the two players chooses to play in that game. Of course, the discussion of this section still holds for small game value intervals (cold games).

CONCLUSIONS AND FUTURE WORK

This paper sketched a human-like learning architecture for the game of *Go*, called HUGO, but it uses no game specific knowledge, so it should be possible to apply it to any two-player, full information, deterministic, combinatorial game. HUGO has three major components. The first component deals with the definition and choice of subgames. The second component is concerned with initiative. In the third component, game values are computed with a generalized α-β algorithm based on fuzzy, partial ordering.
The notion of GOTE NO SENTE (a move that loses initiative but creates new lines of play that will hold initiative) is formalized for the first time.
There are some points where one can apply further machine learning techniques, for instance a clustering algorithm in the extended α-β algorithm. Furthermore, each component has some valuable control parameters. The iterative widening in component 2 and possible iterative deepening in component 3 could further enhance real-time behaviour. The value of uncertainty can be used to control the style of play along a safe-risky axis. Currently, work is carried out to mathematically formalize and implement the α-β algorithm based on fuzzy partial ordering, having combinatorial game values with uncertainty as output.

REFERENCES

Berlekamp, E.; J.H. Conway; and R.K. Guy. 1982. *Winning Ways (for your mathematical plays).* Academic Press, New York.

Cazenave T. 1996. *Systeme d'Apprentissage par Auto-Observation. Application au Jeu de Go.* PhD thesis, Université Pierre et Marie Curie, Paris.

Conway J.H. 1976. *On Numbers and Games.* Academic Press, New York.

Müller M. 2000. "Partial Order Bounding: A new Approach to Evaluation in Game Tree Search." Technical Report of ETL, TR-00-10. To appear in a special issue on heuristic search of the Artificial Intelligence Journal.