Strategic Evaluation in Complex Domains

Tristan Cazenave

LIP6
Université Pierre et Marie Curie
4, Place Jussieu, 75005 Paris, France
Tristan.Cazenave@lip6.fr

Abstract

In some complex domains, like the game of Go, evaluating a position is not simple. In other games, like Chess for example, material balance gives good and fast to compute insight on the value of a position. In Go all the stones have the same value, so material balance is not a good heuristic. To evaluate a Go position, a computer needs a lot of knowledge and much more time. Evaluation in computer Go is interesting from an AI point of view, because it shows the power of knowledge in complex and real world domains.

Introduction[1]

Evaluation functions are usually quite simple and fast. The simplicity of evaluations functions enables to concentrate on the search algorithm, and to replace the knowledge used by humans to solve problems by intensive search. Many researchers have recognized that there is a search vs. knowledge tradeoff [Michie 1977] [Berliner & al. 1990] [Junghanns & Schaeffer 1997]. However in some domains like the game of Go, simple, fast and good evaluation functions do not exist (or at least have not been found despite a lot of efforts). Evaluating positions in such domains requires some times and a lot of knowledge. These domains are interesting for AI because they show the power of knowledge over brute force. They enable to devise, test and compare AI techniques related to the acquisition, learning, management and use of different types of knowledge [Pitrat 1990]. Finding a way to use knowledge so as to be efficient in these complex domains will also advance the state of the art of domains where search is important by improving search with knowledge. This is a more general approach to problem solving, this is the one humans use [McCarthy 1997].

In the first part we present the interest of the game of Go from an AI point of view. Then, we present our method to evaluate positions. In the following part, we show our this evaluation is integrated into a Go playing program.

Computers and the game of Go

The game of Go

Go was developed three to four millennia ago in China; it is the oldest and one of the most popular board game in the world. Like chess, it is a deterministic, perfect information, zero-sum game of strategy between two players. In spite of the simplicity of its rules, playing the game of Go is a very complex task. [Robson 1983] proved that Go generalized to NxN boards is *exponential in time*. More concretely, [Van den Herik & al. 1991] and [Allis 1994a] define the *whole game tree complexity* A. Considering the average length of actual games L and average branching factor B, we have $A = B^L$. The *state-space complexity* of a game is defined as the number of legal game positions reachable from the initial position of the game. In Go, L≈150 and B≈250 hence the game tree complexity $A \approx 10^{360}$. Go state space complexity, bounded by $3^{361} \approx 10^{172}$, and game tree complexity are far larger than those of any other perfect-information game. Moreover, a position is very difficult to judge, on the contrary of chess where a good heuristic for evaluating a position is the material balance. This makes Go very difficult to program.

Computer Go

As searching deep enough is not possible for the game of Go, the best Go playing programs rely on a knowledge intensive approach. They are generally split into two parts:

- A tactical module that develops narrow and deep search trees. Each tree is related to the achievement of a goal of the game of Go.
- A strategic module that chooses the move to play according to the results of the tactical module.

We will focus on the strategic module that takes into account the global position to evaluate. Concerns about evaluating global positions in the game of Go appeared in

[Fotland 1993], where fuzzy status of groups were used to make strategic decisions. [Bouzy 1995] developed further the strategic part involved in Go programs and managed relations between groups with fuzzy status. [Cazenave & Moneret 1997] gives a method to develop strategic plans in situations involving uncertainty.

Evaluating a position

Strategic knowledge in games is about long term goals. In games such as Chess and Go, the high number of possible moves makes it impossible to forecast in the long term the consequences of the moves played. A solution to this problem is to have a gradual achievement of long term goals. It enables to know if a move makes the goal easier or harder to achieve. There are mainly two ways of managing a complex situation, breaking the problem into subproblems and relax the problem by defining a gradual achievement of it.

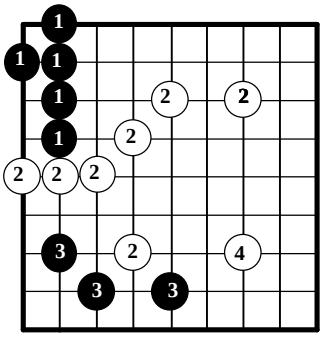

Figure 1

This is particularly true for the strategy in the game of Go. The ultimate goal of a player is to make live the more stone on the board. However, in the middle game, most of the groups of stones (a group of stones is a set of stones of the same color which cannot be disconnected, stones of the same group have the same number in Figure 1) are in an uncertain state, and the evolution of this state cannot be precisely foreseen. It is very useful in such a case to have a gradual evaluation of their states and of the evolution of this state when playing different moves.

A friend intersections of a group is an empty intersection that can be connected to the group whatever the opponent plays, moreover, this empty intersection must not be connectable to a living opponent group.

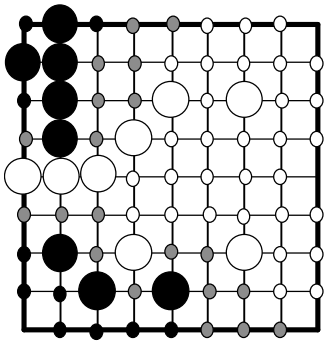
Figure 2

In Figure 2, the white friend intersections are filled with a small white point. The black friend intersections are filled with a small black point. The intersections thatcan be connected both to a white and a black group are filled with a small gray point. Each group owns a set of friend intersections of its own color.

The number of friend intersections of a group is a very good heuristic to approximate the degree of life of a group. For example, the group marked with 2 in Figure 2 has more than twelve friend intersections, it will therefore have no problems to live. Whereas the group marked with 3 in Figure 2 has only 7 friend intersections, it is not completely alive and may have some problems. Its degree of life is around 0.5. Two rules define the degree of life of a group given its number of friend intersections:

```
Degree_of_life ( N, G, F ) :-
        Number_of_friend_intersections ( N, G, H ),
        H > 3,
        F1 =  ( H - 3 ) / 9,
        F = min ( F1, 1.0 ).

Degree_of_life ( N, G, F ) :-
        Number_of_friend_intersections ( N, G, H ),
        H < 4,
        F = 0.0.
```

After these rules have been fired, one rule chooses the greatest of all the degrees of life.

The gradual degree of life is given by the real number F, the group is represented by the variable G, and the integer N is the number of moves to play to achieve this degree of life. The Figure 3 gives the graphical representation of the gradual achievement defined by the rules above.

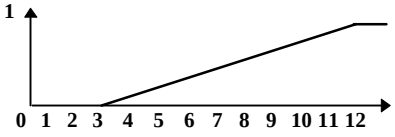

Figure 3

Many predicates contributes to the final goal of the game: having more living stones than the opponent. These contributions are more or less graduals. They are represented in Figure 4. The vertical axis always represents the degree of life of the group, between 0 and 1.

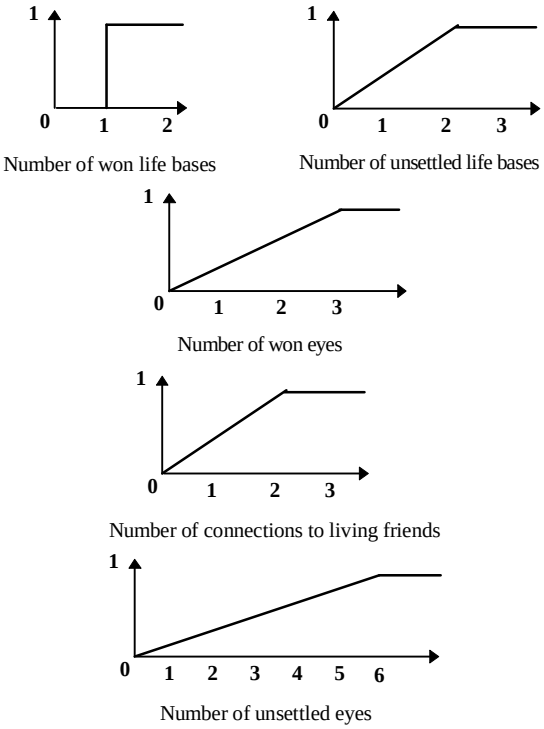

Figure 4

Table 2 gives an evaluation of the attributes for the four groups of Figure 1.

Attributes\Groups	1	2	3	4
Number of won life bases	0	0	0	0
Number of unsettled life bases	1	0	0	0
Number of won eyes	1	0	0	0
Number of unsettled eyes	1	0	0	0
Number of friend intersections	3	26	7	11
Number of stones	5	7	3	1
Number of connections to living friends	0	0	0	2

Table 2

Table 3 gives the degrees of life corresponding to each attribute for each group and also gives the final degree of life for the groups.

Attributes\Groups	1	2	3	4
Number of won life bases	0	0	0	0
Number of unsettled life bases	0.5	0	0	0
Number of won eyes	0.33	0	0	0
Number of unsettled eyes	0.16	0	0	0
Number of friend intersections	0	1	0.44	0.8
Number of connections to living friends	0	0	0	9 1
Degree of life of the group	0.5	1	0.44	1

Table 3

The strategic evaluation function in a Go playing program

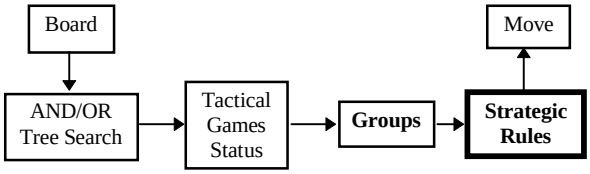

Figure 5

Our Go playing program is named Gogol. It develops AND/OR tree searches to calculate the states of tactical games. Each tactical game corresponds to a simple subgoal of the game of Go. The tactical games status are used to create the groups and to fill the predicates used by the strategic module. Gogol develops approximately 1000 proof tree searches on a position. It develops trees using Proof Number Search [Allis & al. 1994b], the result of a tree is a tactical theorem that applies to the board at hand: the moves advised by the theorem always reach the tactical goal used during the search. These proof trees contain between 2 and 600 nodes. Once the tactical results are deduced, the program fires the strategic evaluation rules that evaluate the degree of life of each group and its evolution after each interesting move. This information is used to choose the best move. The best move is chosen by evaluating the difference of the board value after and before each move. The best move is the move that has the highest difference.

To evaluate the value of the board, the system has to evaluate the degree of life and the importance of each group. The importance of a group is the evaluation of the difference of points at the end of the game between the life of the group and its death. It is computed using the following rule:

```
Value ( G, N ) :-
        Number_of_stone ( G, N1 ),
        Number_of_friend_intersections ( G, N2 ),
        Number_of_shared_friend_intersections ( G, N3 ),
        N = N1 + N1 + N2 + N3.
```

Groups	1	2	3	4
Value of the group	24	80	32	31

When the values and the degrees of life of the groups have been computed, the system can evaluate a Go board:

Evaluation =

$$\sum_i (\text{Degree}_i * \text{Value}) - \sum_j (\text{Degree}_j * \text{Value})$$

with $i \in$ Friends Groups and $j \in$ Opponent Groups.

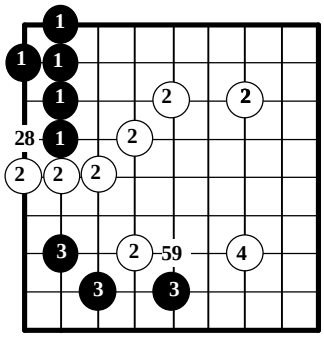

Figure 6

In the example of Figure 2, if black is the friend color, the evaluation of the position gives:

Evaluation=0.5*23+0.44*32-1.0*80-1.0*31=-85.4

This evaluation means that black is probably going to lose the game by 43 points. This analysis is compatible with the analysis of Go expert players. This evaluation function has been tested on numerous Go boards and it gives a good approximation of the evaluation of a position.

The two moves we are examining in the board of Figure 6 are the black moves at i28 and i59. Table 4 gives the outcomes of the black move at i28 and Table 5 gives the outcomes of the black move at i59.

Attributes\Groups	1	2	3	4
Number of won life bases	+1	0	0	0
Number of unsettled life bases	-1	0	0	0
Number of won eyes	+1	0	0	0
Number of unsettled eyes	-1	0	0	0
Number of friend intersections	0	0	0	0
Number of connections to living friends	0	0	0	0

Table 4

Attributes\Groups	1	2	3	4
Number of won life bases	0	0	0	0
Number of unsettled life bases	0	0	0	0
Number of won eyes	0	0	0	0
Number of unsettled eyes	0	0	+1	0
Number of friend intersections	0	-4	0	-1
Number of connections to living friends	0	0	0	-1

Table 5

If the board is evaluated after the two black moves, there is a variation of +12 points for the black move at i28 and a variation of +11 points for the black move at i59. The system will choose the black move at i28.

Results

The best Go programs are those that have the best strategic evaluation function and the most precise tactical search engines. But it takes times to evaluate position, because for each strategic position evaluation, a lot of tactical searches have to be performed. So Go programs cannot search very deep at the strategic level. The precision of the evaluation function is therefore very important. It is based on a good knowledge of what are the important concepts of the game of Go (such as territory, influence, groups and their degrees of life).

Gogol plays a move in 10 seconds on a Pentium 133 MHz. It has participated in the 1997 FOST cup held during IJCAI97. It has finished 6 out of 40 participants. The five first programs are commercial programs.

Future work is to use learning, as described in [Cazenave 1996], at the strategic level. The goal of learning will be to improve the evaluation of positions and to find strategic moves interesting to try.

Conclusion

Evaluation in computer Go is interesting from an AI point of view, because it shows the power of knowledge in complex and real world domains. In the search versus knowledge tradeoff, the game of Go is the one that has the most important knowledge component. We have shown how a complex evaluation function can be devised by breaking the problem into subproblem, and relaxing the goals by making them gradual. This approach has been used to write the evaluation function of a Go playing program. It has shown its usefulness during the last FOST cup [Fotland 1997], an international competition between Go programs.

References

Allis, L.V. 1994a. *Searching for Solutions in Games and Artificial Intelligence*, Ph.D. Thesis, Vrije Universitat Amsterdam, Maastricht, September 1994.

Allis, L.V.; Meulen, M. van der; Herik, H.J. van den 1994b. Proof-Number Search. Artificial Intelligence, Vol. 66, No. 1, pp. 91-124.

Berliner, H.; Goetsch, G.; Campbell, M.; Ebeling, C. 1990. *Measuring the performance of potential chess programs*. Artificial Intelligence, 43(1) :7-21, April 1990.

Bouzy, B. 1995. *Modélisation cognitive du joueur de Go*. Thèse de l'université Paris 6, 1995.

Cazenave, T. 1996. *Système d'Apprentissage par Auto-Observation. Application au Jeu de Go*. Thèse de l'Université Paris 6, Décembre 1996.

Cazenave, T.; Moneret, R. 1997. *Development and Evaluation of Strategic Plans*. Game Programming Workshop'97, Hakone, Japan 1997.

Fotland, D. 1993. *Knowledge Representation in The Many Faces of Go*. Second Cannes/Sophia-Antipolis Go Research Day, Février 1993.

Fotland, D.; Yoshikawa, A. 1997. *The 3rd fost-cup world-open computer-go championship*. ICCA Journal 20 (4):276-278.

Junghanns, A.; Schaeffer, J. 1997. *Search Versus Knowledge in Game-Playing Programs Revisited*. IJCAI97 p. 692-697, Nagoya, Japan, 1997.

McCarthy, J. 1997. *Review of Monty Newborn's Kasparov versus Deep Blue : Computer Chess Comes of Age*. Science , 6 June 1997.

Michie, D. 1977. *A theory of advice*. Machine Intelligence 8, p. 151-170, 1977.

Pitrat, J. 1990. *Métaconnaissances. Futur de l'Intelligence Artificielle*. Editions Hermes, Paris, 1990.

Robson, J. M. 1983. *The Complexity of Go* - Proceedings IFIP - pp. 413-417 - 1983.

Van den Herik, H. J.; Allis, L. V.; Herschberg, I. S. 1991. Which Games Will Survive ? Heuristic Programming in Artificial Intelligence 2, the Second Computer Olympiad (eds. D. N. L. Levy and D. F. Beal), pp. 232-243. Ellis Horwood. ISBN 0-13-382615-5. 1991.