Complex Games in Practice

Bruno Bouzy

C.R.I.P.5
UFR de mathématiques et d'informatique
Université René Descartes (Paris V)
45, rue des Saints-Pères 75270 Paris Cedex 06 FRANCE
tel: (33) (0)1 44 55 35 58 fax: (33) (0)1 44 55 35 35
e-mail: bouzy@math-info.univ-paris5.fr
http://www.math-info.univ-paris5.fr/~bouzy/

keywords :

Game of Go, Group Strength, Monocolor Tree Search, Killing and Living Moves Numbers

1. Introduction

This paper highlights an important issue linked to the global move decision process in a Go playing program, in other terms, how to describe, determine and use group strength in a simple way so as to select the move to play. Our present study is based on our experience in developing Indigo, a current Go program that ranked tenth at the last Ing Cup in London in November 1998. Therefore, this contribution mainly relies on practical considerations, not theoretical ones.

After a short preliminary section, this paper deals with the dynamic of group strength, which is a very crucial subject for computer Go. We introduce two numbers, the Killing Moves Number (KMN) and the Living Moves Number (LMN), in order to define a " complex " game in practice. We compare our approach with [Conway 76] and two practical approaches [Tajima & Sanechika 98] and [Cazenave 96]. The following section focuses on Monocolor Tree Search (MTS). We think MTS is a good complement to adverse tree search. Then, within the global move decision context, the notions of urgency, order/disorder are discussed. We hope our paper will help uncover some crucial points of Computer Go and explicit some features not to be found in other games.

2. Assumptions

After a quick description of territory evaluation, we will show how important it is for Computer Go to assess the strength characteristic of the different groups.

2.1. Quiet positions and territory

By means of a simple diffusion model [Zobrist 69] or mathematical morphology [Serra 82], a program recognizes the territory more or less accurately on a quiet position. Therefore, by simply searching for each color at depth one, the program provides a series of switches and thus selects the switch with the hottest temperature. Through such a method, the program can expand or reduce territories in a satisfactory way. Of course, this method is time-consuming but this can be easily reduced by only evaluating moves selected by some patterns. Another time improvement can be performed by memorizing as many switches as possible from one position to the next. Unfortunately, this method has a weak point : it completely ignores groups and it will always prefer scoring territories to attacking or defending groups. At that stage, a model of groups is required.

2.2. Static description of the group strength and its difficulties

The difficulties lie in defining groups correctly… The groups can be designed with an influence model

[Zobrist 69] [Boon 91] [Chen 99], which corresponds to the players' intuitive view. The groups can also be designed thanks to a set of connection patterns [Bouzy 95] [Cazenave 96]. Both approaches can occasionally be used. In this paper, we assume the definition of groups is given. Whatever the definition, we think another difficulty arises from modelling the group state. This model must embrace live groups, dead groups and " in-between " groups, thus offering many possibilities. Concerning live groups, we may adopt two different approaches : a restrictive and clear one (two eyes or seki) or a recursive and fuzzy one (same conditions, plus capturing an adjacent enemy group, plus connecting to friendly live groups, plus escaping toward some empty space). The restrictive definition has the advantage of precision. It avoids making serious mistakes in global evaluation and can be integrated into a tree search. The second definition is recursive (it uses the state of nearly groups) and therefore is difficult to handle. Morever, it is fuzzy because it uses empty space which is hard to define with accuracy. In the very end it is time-consuming. But it has the advantage of corresponding to the players' description in real games. Unless proper tuning is achieved, this approach may cause big blunders. In our current study, we do not intend to describe the static properties of a group. This is too difficult a task and it greatly depends on the programmer's capacity to analyze the nature of Go. Instead, we assume that the description of group strength is given by a function returning three possible values : DEAD, OTHER, ALIVE for each group, GSEF for Group Strength Estimation Function.

3. Group strength estimation is a " complex " game in practice

In this section, we will take up the assumptions previously stated so as to describe what happens concerning the dynamic of groups. Before defining what a " complex " game can be in practice, we have decided to adopt two relevant numbers, the Killing Moves Number (KMN) and the Living Moves Number (LMN).

3.1. Static classification

Figure 1 shows the three states of a group :

Figure 1

Given such a classification provided by the GSEF, the meaningful moves are the jumps from one state to another. The moves that do not correspond to any jump are irrelevant for the program.

3.2. Dynamic classifications with Adverse Tree Search (ATS)

In our present paper, we call Adverse Tree Search (ATS), a tree search involving alternative colors at each move. This is tree search in its common use. ATS offers the possibility to refine the " OTHER " state (Figure 1) into the following three substates : " dead ", " unknown " and " alive " (Figure 2) :

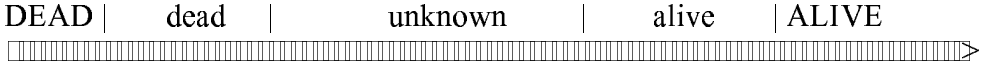

Figure 2

ATS detects the groups in the " OTHER " state of Figure 1 that cannot avoid either the " DEAD " state, or the " ALIVE " state, whoever plays first. The " dead " or " alive " states of Figure 2 corresponds to a group which is not dead or alive according to the GSEF, but whoever plays first, the group will die or live.

Therefore we can merge the states " DEAD " and " dead " of Figure 2 into one single " {DEAD} " state (see Figure 3 below). The same can be done for the state " {ALIVE} ".

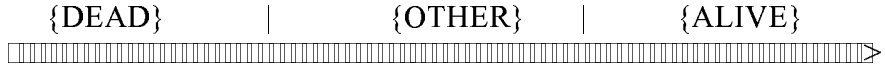

Figure 3

ATS enables the program global level to use transitions {OTHER} ⇨ {DEAD} or {OTHER} ⇨ {ALIVE} that were not mentioned in Figure 1. ATS also eliminates transitions OTHER ⇨ DEAD or OTHER ⇨ ALIVE in Figure 1. Classification in Figure 3 is much more reliable and useful than classification in Figure 1. ATS prevents the program from playing irrelevant moves by selecting efficient ones. When ATS increases, the states {DEAD} and {ALIVE} become bigger (see Figure 4 below).

Figure 4

As a result, our classification is modified and shows similarities with Conway's classification (see Figure 5).

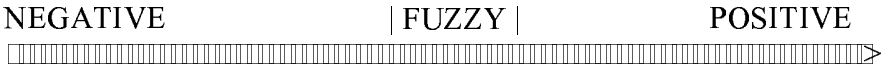

Figure 5

But let us go back to practice and Figure 3. In order to underline the fact that ATS has tried to prove a group was dead or alive unsuccessfully we delete the brackets and the word UNDETERMINED will be used to define the {OTHER} state.

3.3. The limits to ATS ; Stability " in practice "

The next problem concerns the cost of ATS. When ATS ends successfully, several nodes have been investigated and time has been swapped for relevant information. But, because of time limits, ATS may not end successfully. In such a case, time loss is almost gratuitous and we believe ATS may not be well adapted to the problem of group strength. When ATS ends unsuccessfully, time loss may seem gratuitous to a certain extent only : the program knows it cannot reach a definite conclusion. By simply observing that if ATS cannot prove a group is dead or alive, it means the group has got a ***kind of stability in practice*** : the group stays in the UNDETERMINED state whoever plays first. This remark leads to a fundamental question : what must the program global level do with such " stable " groups ? One may leave them as they are and play elsewhere or one may add moves until a real stability is reached (i.e. state DEAD or state ALIVE). These two options have their own advantages. Playing elsewhere avoids making unnecessary moves but the risk is to see these groups going into the wrong final state in a sudden and unexpected way. Playing on these groups avoids leaving uncertainty on the board but the risk is to follow an unreachable goal and/or to play unnecessary moves. We think no good solution can be found without measuring what happens within the UNDETERMINED state.

3.4. A measure within the UNDETERMINED state

The GSEF is more accurate when some expertise is brought. GSEF returns more precise information by splitting the UNDETERMINED state into substates :

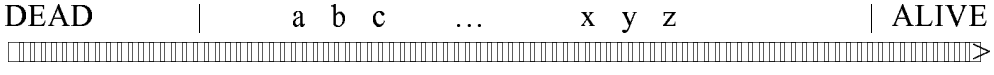

Figure 6

By means of this measure, the program uses efficient transitions [Tajima & Sanechika 98] [Bouzy 96], hence the moves become meaningful. As a first clue, the group that stays in the middle of the UNDETERMINED can make steps in order to reach a final state. But the quest for finding a sound measure is still open [Fotland 96]. [Chen 99] or [Boon 99] seem to have a sound measure because their

programs use TS to build one switch for each group.

3.5. The " LMN " and " KMN " numbers

When no correct measure of group strength is available, one simple measure consists in ***counting the number of moves of the same color in order to kill or save the group***, that is the number of friendly moves to get out of the UNDETERMINED state into the ALIVE state or the number of enemy moves necessary to reach the DEAD state. The classification is now as follows :

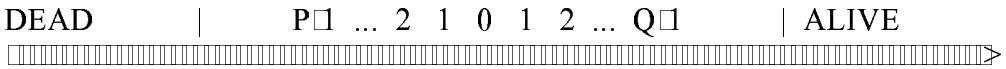

Figure 7

P corresponds to the Killing Moves Number (KMN) and Q to the Living Moves Number (LMN). A group that stays in the UNDETERMINED state will be represented with the following notation : **[P|Q]**.

3.6. A " complex " game

In our paper, we shall call ***complex*** a game where KMN and LMN are big enough to prevent ATS from succeeding. According to us, a complex game is a game during which ATS fails and other techniques are needed. To this extent, the game of Go is of course complex and so is the group strength sub□game. The group strength sub□game, also named the generalized life and death sub□game in the litterature, leads to important obstacles [Wolf 96]. Furthermore, when KMN and LMN become low, the group strength sub□ game becomes the classic life and death problem every Go player is familiar with and which is successfully addressed by GoTools [Wolf 94].

3.7. Important transitions

Of course, our discussion must not overlook that the transitions near the ALIVE or DEAD states (P or Q equals 1 or 2) are still more important than intermediate transitions (when P or Q are greater than 3). Figure 8 links important transitions to the commonly used terms in the world of games :

from ALIVE□1 to ALIVE	a ***terminal*** move
from ALIVE□2 to ALIVE□1	a ***threat*** to reach the friendly goal
from DEAD□1 to DEAD□2	a ***forced*** move to avoid the enemy goal

Figure 8

3.8. Related studies

Comparison with [Conway 76]

The difference between Figure 5 and Figure 3 shows the difference between theory and practice. In theory, when you play a move from the FUZZY state of [Conway 76], you are supposed to reach either the POSITIVE or NEGATIVE state in one single move. In practice, the information about the game is not complete and you may spend many moves before getting out of the UNDETERMINED state. This is the reason why we split the intermediate state into several sub□states. Conversely, Conway's numbers correspond to our ALIVE or DEAD states. These two final states could be split into sub□states so as to follow Conway' numbers but, at the moment, this possibility remains a promising perspective in our work.

Comparison with [Tajima & Sanechika 98]

[Tajima & Sanechika 98] has recently suggested PON, Possible Omission Number, as the number of moves a group can bear without being killed. First, let us make the things clearer with Figure 9 below. The first two lines corresponds to our classification, the third line to PON and the fourth line to [Tajima

& Sanechika 98] 's terminology.

DEAD				UNDETERMINED					ALIVE	Bouzy
DEAD		P□1	P□2	…	1	0	1	…	Q□1 ALIVE	KLMN
□N … □2	□1	0	1	…	P□2	P□1	P	...	P+Q□2 N	PON
DEAD	NEUTRAL	ALIVE	…	ALIVE		…			ALIVE	T&S

Figure 9

The definition of PON corresponds to numbers in [Conway 76]. If GSEF is well□tuned, we obtain FUZZY = NEUTRAL. [Tajima & Sanechika 98] classification assumes that the GSEF determines the life of a group with precision. For example, an eyeless group which is not surrounded enjoys an ALIVE status in [Tajima & Sanechika 98]. On the contrary, our GSEF will classify the same group as belonging to the UNDETERMINED state. We use such an approach because the task of statically determining the life of groups which are not surrounded is very difficult. But of course, the decision to have a simple GSEF or a sophisticated one depends on its integration with the other program components.

Comparison with [Cazenave 96]

[Cazenave 96] has proposed an explicit representation of uncertainty inherent in Go sub□games by using a taxonomy of states very useful in practice. The most general state of taxonomy is the state " I " for a game whose left and right outcomes are not known (" I " stands for " Incertain " which means uncertain in French). So as to represent these two uncertain outcomes, this game should also be defined as " II ". A game " IP " is a game where Right may make the game lost for Left in one move and where one does not know about Left outcome. A game " IIIP " is a game where Right can have the game lost by playing two moves in a row and where the other three outcomes are not known. Symmetrically, the game " GI " is a game where Left can win in one move and where the right outcome is not known. The game " GP " is equivalent to the game * in [Conway 76]. This classification offers an explicit representation as regards the lack of information. This is very useful in practice because the computer cannot always complete a full TS.

[Cazenave 96] did inspire our present work. Cazenave's classification was applied to elementary Go sub□ games such as eye formation and connection between strings. Therefore, this work mainly focused on games such as IP, IIIP, GI, GIII, GP, GIP but not much on GIII…IIIP games (with a lot of I). [P|Q] in our terminology corresponds to GIII…IIIP (with P times the letter I on the left and Q times the letter I on the right) in [Cazenave 96]. Therefore, our notation with KLMN is simpler.

DEAD	P□1	P□2	…	0	…	Q□1	ALIVE	KLMN
P	IP	IIIP	…	GIII…IIIP	...	GI	G	Cazenave

Figure 10

Conclusion

We can say that, when the program imposes very restrictive conditions for life and death, the pair of KLMN numbers is a good way to describe a group which stays in the UNDETERMINED state. One can choose to use PON when the program already uses a well□tuned GSEF. The FUZZY state [Conway 76] is not useful in most practical cases that arise when estimating group strength. We think using KLMN numbers or Cazenave's classification can be better in practice because Go programs have so far failed to be strong.

4. Monocolor Tree Search (MTS)

We call Monocolor Tree Search (MTS) a search during which only one player only is making several

moves in a row in order to reach a goal. MTS is very useful because it enables the program to find KLMN by selecting efficient moves (i.e. a set of moves that drive quickly to the ALIVE or DEAD states). We have identified three possible uses of MTS in a Go playing program : goal selection and move selection at strategical level and move generation within ATS.

4.1. Goal selection at strategical level

As regards goal selection, MTS can be used appropriately. Provided you can estimate the gain G obtained by reaching the goal, you can compute a rentability R attributed to the goal with :

R = G / KMN or R = G / LMN.

4.2. Move selection at strategical level

By definition, MTS selects the most efficient moves. Then, at strategical level, the problem is to choose moves among a set of moves with different KLMN, an issue to be discussed in the next paragraph □move selection at the strategical level has similarities with move generation within ATS.

4.3. Move generation within ATS

When KLMN are small enough (equal 2 or 3) the moves which are found by MTS help the ATS move generation. To this extent, ***MTS might be considered as a preprocessing to ATS***.

During ATS, the program reaches some positions where evaluation is meaningful. On such positions, the group states are either DEAD or ALIVE but not UNDETERMINED. More generally, when evaluation is meaningful, it is composed of sub□games whose states are either WON or LOST but not OTHER. Under time constraints, ATS must be guided toward such quiet positions and sub□game terminal moves are able to drive the global search toward these calm positions. Therefore, sub□game terminal moves must have a high urgency within the move generation process. Terminal moves establish order inside the positions. Consequently, we call them, ***ordering moves***.

On the other hand, the sub□game forced moves are ***disordering moves***. First, by definition, they erase the terminal ordering moves associated to the sub□game, second, they lead to the middle area of the UNDETERMINED state, impossible to evaluate with certainty, third, they may generate other sub□games with UNDETERMINED states and thus, still increase the uncertainty of the evaluation.

Threats have an ordering quality because they guide the search toward the WON state which is an ordered state. But this ordering quality is weak because such moves will generate both ordering moves (at least one terminal move) and disordering ones (at least one forced move) at the next position. Therefore, we call such moves, ***weak ordering moves***.

Moves that correspond to the WON□>WON□1 transition are purely ***disordering moves***. (Such moves exist because a game may be virtually but not actually won ; see for example, a " ladder breaker " or an " atari " on a safe string). They destroy the order of a terminal state (P or Q = 0) and create a new state with uncertainty (P or Q > 0).

In summary, we propose a decreasing urgency for these 4 move classes that reflects their ability, on the one hand, to establish order inside the position and, on the other hand, to decrease the position uncertainty:

Urgency(terminal move)
> Urgency (threat)
> Urgency (forced move)
> Urgency (WON□>WON□1)

Of course the above urgency must be refined with a factor that depends on the ***nature*** of the sub□game. For example, the urgency of a terminal move linked to a capturing string sub□game will not be the same

as the urgency of a terminal move linked to a connection sub-game. At the very end, the urgency will also depend on the ***size*** of the sub-game, that is the size of the group for a group strength sub-game or the size of the string for a capturing string sub-game.

4.5. Implementing MTS in Indigo98

MTS Implementation was part of Indigo98. Before depth 3, MTS is not very consuming. For instance, finding terminal moves is easy. Finding threats takes more time because the program must verify that a terminal move will follow the threat. Finding forced moves is still longer, because the program must find at least one terminal move for the enemy, then try a possible friendly move and verify that there will be no ensuing terminal move for the enemy. MTS beyond depth 3 was not implemented because of time constraints. To us, this approach seems to be a cost-effective way to find promising moves in the group strength sub-game without using the very expensive ATS.

5. Conclusion

In this paper, we have studied the group dynamic in the game of Go within a practical framework. A static classification (DEAD, OTHER and ALIVE) was assumed to be provided by an Group Strength Estimation Function (GSEF). Adverse TS (ATS) applied to the group strength sub-game was time-consuming in the wrong case and may not be well-adapted. Consequently, we suggested two numbers " KMN " (for Killing Moves Number) and " LMN " (for Living Moves Number) corresponding to the number of moves to be played in a row so as to kill or save the group. We defined a " complex " game in practice as a game where KMN or LMN are high. We presented the " Monocolor Tree Search " (MTS) to compute KMN and LMN. We compared our contribution with the G, I, P classification of [Cazenave 96], with the Possible Omission Number (PON) of [Tajima & Sanechika 98] and with the Fuzzy, Positive, Negative, Zero classification of [Conway 76]. We identified three possible uses of MTS : affecting a rentability to goals that are selected at strategical level, selecting promising moves at strategical level and preparing ATS move generation. Finally, we discussed the notion of urgency when P and Q are small (equal 1 or 2) in the context of the global move decision context.

We think that, in the near future, practical studies on complex games like the group strength sub-game in Go will enhance the importance of simple tools and concepts such as KLMN or MTS.

6. References

[Boon 91] Mark Boon, ***Overzicht van de ontwikkeling van een Go spelend programma***, Afstudeer scriptie informatica onder begeleiding van prof. Bergstra J, Amsterdam, 1991.

[Boon 99] Mark Boon, message sent to the computer-go@hsc.fr mailing list on May 17th 1999.

[Bouzy 95] Bruno Bouzy, ***Modélisation cognitive du joueur de Go***, thèse de l'université Paris 6, 13 janvier 1995, http://www.math-info.univ-paris5.fr/~bouzy

[Bouzy 96] Bruno Bouzy, ***There are no winning move except the last***, Proceedings of the 6th International Conference on Information Processing and Management of Uncertainty in Knowledge Based Systems, IPMU'96, Granada, Spain, July 1-5, 1996.

[Cazenave 96] Tristan Cazenave, ***Système d'apprentissage par auto-observation. Application au jeu de Go***, thèse de l'université Paris 6, 13 décembre 1996. http://www.ai.univ-paris8.fr/~cazenave

[Chen 99] Chen Zhixing, ***Programming Technics in Handtalk***, http://www.wulu.com/, 1999.

[Conway 76] John Conway, ***On Number And Games***, Academic Press, 1976.

[Fotland 96] David Fotland, ***Computer Go Design Issues***, Message sent to the computer-go@hsc.fr mailing list on October 1st 1996.

[Tajima & Sanechika 98], Morihiko Tajima, Noriaki Sanechika, ***Estimating the Possible Omission Number for Groups in Go by the Number of nth Dame***, , First International Conference on Computer and Games 98, in Lecture Notes in Computer Science, n° 1558, H.J. van den Herik, Hiroyuki Iida (eds), pp. 265-281, Springer.

[Serra 82], Jean Serra, ***Image analysis and mathematical morphology***, Academic Press, London, 1982.

[Wolf 94], Thomas Wolf, ***The program GoTools and its computer-generated tsume-go database***. First Game Programming Workshop in Japan, Hakone, 1994.

[Wolf 96], Thomas Wolf, ***About problems in generalizing a tsumego program to open positions.*** Game Programming Workshop in Japan'96, Hakone, 1996.

[Zobrist 69], A Zobrist, ***A model of visual organisation for the game of Go***, Proceedings AFIPS 34, pp103-112, 1969.