当年“手谈”程序的文章

ELSEVIER

Information Sciences 121 (1999) 113–134

INFORMATION SCIENCES
AN INTERNATIONAL JOURNAL

www.elsevier.com/locate/ins

Static analysis of life and death in the game of Go

Ken Chen [a,*], Zhixing Chen [b]

[a] *Department of Computer Science, University of North Carolina, Charlotte, NC 28213, USA*
[b] *Department of Chemistry, Zhongshan University, Guangzhou, People's Republic of China*

Received 18 February 1998; received in revised form 13 December 1998; accepted 20 February 1999

Abstract

This paper describes heuristic rules for static life/death analysis for general classes of groups in the game of Go, as implemented in authors' world championship computer Go programs Go Intellect and HandTalk. We shall show that for many life and death problems in Go, the effect of a tree search can be approximated by a static analysis on the group's interior and surroundings. We investigate eye-spaces with or without opponent's dead stones and try to provide a foundation towards conquering the combinatorial explosion of Go game tree. © 1999 Elsevier Science Inc. All rights reserved.

Keywords: Heuristic programming; Approximation algorithms; Machine game playing; Combinatorial game theory

1. Introduction

Recent success of the Deep Blue chess program draws vast attention and excitement to machine game playing. Yet the ultimate programming challenge in machine game playing undoubtedly lies with the ancient oriental game of Go. It is well known that Go defies computer chess type full board search techniques.

Go is a two-person perfect information game. Two players alternately place a black and a white stone on some empty intersection of a 19 by 19 grid. Placed

* Corresponding author.

stones are never moved, but may be removed (called captured or killed) if they are completely surrounded. Basically every empty intersection is a legal move point. The rare exceptions are the points that playing a stone would cause an earlier board configuration in the game to reoccur. They are referred as 'ko' situations. The objective of Go game is to secure more territory (grid points) than the opponent does. Kierulf et al. [6] contains a lucid explanation of Go game and the task of building a Go program. For more detailed descriptions of the game and the rules of Go, readers can visit American Go Association web site. Its home page address is http://www.usgo.org/ and its AGA Rules of Go page is at http://www.cs.cmu.edu/People/wjh/go/

A Go game normally runs over 200 moves. Each turn offers about 250 choices of legal plays on average. Because of this high branching factor and difficult positional understanding, Go is considered a most challenging game for the machine. It provides an excellent model for exploiting heuristic programming techniques in artificial intelligence.

The determinations of Life/death for groups on the board are fundamental tasks in Go games. Board evaluation, candidate move generation, and move decision making all rely on the life/death knowledge of groups on the board. Unfortunately group life/death problem in Go is just as hard as the whole game of Go. Lichtenstein and Sipser [8] and Robson [11] have shown that life and death problem in nXn Go is *P*-space hard and exponential time complete respectively.

Life/death in Go is a combinatorial problem. It is to determine whether a group can survive under any of the opponent's exponentially many attacking sequences. Or equivalently, it is to find the game theoretic value of a local game tree. The natural approach is of course to perform a thorough tree search and do the mini–max back up. But such approach runs into combinatorial explosion as noted above. And it may not be practical under normal tournament time constraints.

When facing NP-hard problems, computer scientists seek approximation algorithms for real-time solutions. This paper will show that for some problems, an approximation algorithm may be entirely different in characteristics from the original exact algorithm – the approximation of a tree search can be a static analysis on a group's surroundings!

2. Investigations of life and death in Go

Benson [1,2] investigated 'unconditional life' – life without the need of any defense (the group is alive even if it passes on every turn). Popma and Allis [10] proposed the notion of '*X* life' – passing *X* times by the defense, the group can still live. Wolf [12] investigated life and death problems using tree search. Muller [9] introduced an algorithm for recognizing safety under alternating play for some restricted types of Go groups. Landman [7] studied combinatorial game theoretic vales of various eye-spaces using an interesting Bargo model. In this paper, we attempt to provide static life/death analysis for general classes of Go groups without the involvement of 'ko'. Performing tree search for every group to determine its life and death status is too expensive. Both authors use static analysis for Life/Death in their Go programs, Go Intellect (K. Chen) and HandTalk (Z. Chen).

Most of the concepts and rules we derive in this paper will be heuristic in nature so they can cover a wide variety of groups. Even though the heuristic rules described in this paper provide the correct results most of the time, there exist exceptions. If we try to define everything strictly and try to prove rules mathematically, then the life and death problem will become too difficult to solve.

We shall follow the terminology of Benson [1,2] and Chen [4]. A block, also known as string, is a maximal connected set of stones of one color. A group is a collection of closely related blocks plus the empty points under their strong influence and surrounded opponent's dead stones. These empty points are called the group's interior spaces and these opponent's dead stones are called prisoners. A group is alive if it can always prevent being captured, no matter how the opponent, playing first, attacks. Experienced Go players know if a group can make 2 eyes or can manage a co-life (seki), it lives. A group is dead if the opponent can always capture it, no matter how we, playing first, defend. A group is critical if it will be alive if the group side moves first and it will be dead if the opponent moves first. We shall investigate the life/death/critical status of groups via static local analysis in the following sections. Ko requires a global analysis and the situation can become extremely complicated (see [11]), this paper will not deal with it. For those groups involve in ko, we can assume, based on a separate ko-threat analysis, a specific winner for the ko before performing static analysis, or simply revert it to global search.

3. Eye-points

Our static analysis is based on heuristic estimation of a group's eye-making ability. Both empty points and opponent's dead stones can become eyes. We shall investigate empty eye-points first. Prisoner eye-points will be studied in a later section.

An empty point surrounded or nearly surrounded by the player's stones is referred to as a controlled point. Controlled points are usually inaccessible by the opponent. They include points forbidden to the opponent by rules of the game and those points at which entered stones of the opponent can be killed. Practically, it is difficult to judge controlled points according to accessibility of the opponent, as it needs considerable search effort to judge accurately. The concept of interior space points of a group, see [4], is a good static approximation of controlled points.

Controlled points may be full, partial, or false eye-points. Full and partial eye-points are referred jointly as eye-points. False eye-points are not eye-points.

In all examples in this paper, we assume that we take white, and the opponent plays black.

Points *A*s in Fig. 1 are obviously non-eye-points. However, some accessible points to the opponent may become inaccessible after our move; thus they can be turned into eye-points. Points *C*s in Fig. 1 are examples. They are now accessible by the opponent. But they will become inaccessible after we move at *A* on the right. These potential eye-points cannot be neglected and should be treated as partial eye-points. Hence frontier spaces, see [4], are candidates for partial eye-points and should also be examined.

The discrimination among full, partial and false eye-points depends on the opponent's accessibility at the adjacent and diagonal-adjacent points of the controlled point. If the adjacent points are all inaccessible to the opponent, it depends on the status of the diagonal points. Even when a controlled point has an opponent accessible adjacent point, such as *E* in Fig. 1 (its left adjacent is accessible to the opponent), its diagonal points should also be investigated.

A controlled point in line 1 (margin line of the go-board) is a false eye-point if any diagonally adjacent point is occupied by a live opponent-stone. For a point above line 1, it is a false eye-point if two or more diagonally adjacent points are occupied by live opponent-stones. Both points marked by A in Fig. 2 are false eye-points. A controlled point that will inevitably become a false eye-point is also classified as a false eye-point. The point *A* in Fig. 3 has a diagonally adjacent point occupied by opponent and two others accessible to opponent, one of which will inevitably be occupied by the opponent, so *A* is also a false eye-point.

A controlled point is a full eye-point if the opponent's move cannot change it to a false eye-point. A controlled point is a partial eye-point (usually half eye-point) if opponent's move can change it to a false eye-point and our move can change it to a full eye-point.

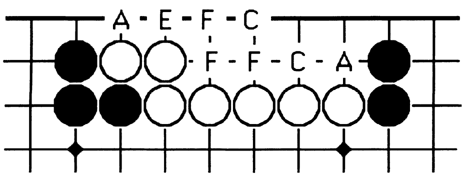

Fig. 1.

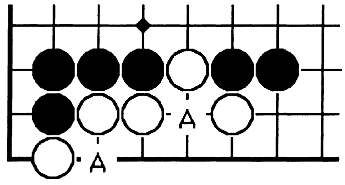

Fig. 2.

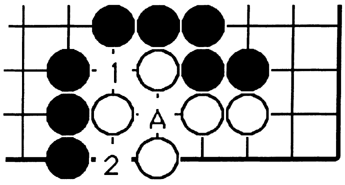

Fig. 3.

A full eye-point is an eye if it is not adjacent to any empty point. A half eye-point is a half-eye if it is not adjacent to any empty point. In Fig. 4 both *E*s are eyes, and *H* is a half-eye.

If the full eye-point condition on diagonals is satisfied, the controlled point has only one empty adjacent point, and the empty adjacent is accessible to the opponent, then this controlled point is a half-eye. Point *D* in Fig. 5 is thus a half-eye.

Therefore, a controlled point is a full eye-point if all adjacent empty points are controlled points and one of the following is true:

1. It is in line 1 and all diagonal points are occupied by our live stones or are our controlled points.
2. It is above line 1 and at least 3 diagonals are occupied by our live stones or are our controlled points.

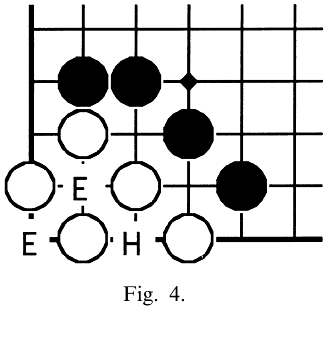

Fig. 4.

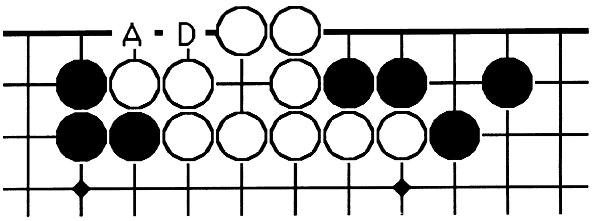

Fig. 5.

3. It is above line 1 and 2 diagonals are occupied by our live stones or are our controlled points and the other 2 are (opponent-accessible) empty points.

A full eye-point is not necessarily able to produce an eye. A full eye-point with an adjacent false eye-point and no other empty adjacent is only a half-eye. In Fig. 6, *D* is a full eye-point with an adjacent false eye-point marked by *A*. If White moves at *A*, the point at *D* is an eye. If Black moves at *A*, White cannot make eye at *D*.

There is no difference between the *D*s in Figs. 5–7 at their diagonal neighbors. The only difference between them is the opponent-accessibility at their left adjacent. Such a difference has no effect on the possibility of eye making. Furthermore, Both *A*s in Figs. 5 and 6 are the same at their diagonal adjacent points. Point *A* in Fig. 6 is a typical false eye-point. Point *A* in Fig. 5 can also be regarded as a false eye-point. We can also regard the *D* in Fig. 5 to be a full

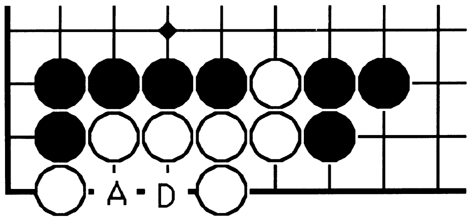

Fig. 6.

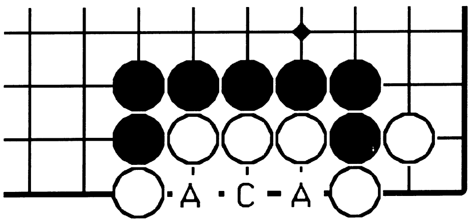

Fig. 7.

Table 1
Initial eye-point classification for points in line 1

Diagonal index	$\leqslant 3$	$3\frac{1}{2}$	4
Classification eye-point	False eye-point	Half eye-point	Full eye-point

Table 2
Initial eye-point classification for points above line 1

Diagonal index	$\leqslant 2$	$2\frac{1}{2}$	$\geqslant 3$
Classification eye-point	False eye-point	Half eye-point	Full eye-point

eye-point in our initial assessment. As a matter of fact, the status of diagonal adjacent is comparatively easy to judge, so the eye-point analysis can be considerably simplified if we initially discriminate full, half, and false eye-points according only to the situation at its diagonal adjacent points. The shortcoming of such a simplification can be remedied by adjustments based on the situations of the adjacent points.

We summarize the above discussion into an efficient discrimination method for our initial assessment. Heuristic eye-point downgrading rules will be introduced in later sections for the needed adjustments. Let call a point on the Go board a controlled point, if it is occupied by our live stone or by opponent's dead stone, or it is a border point or our controlled empty point. An empty board point is said to be a non-controlled point if it is accessible by both sides. The diagonal index of a board point is the number of our controlled board-points, plus 1/2 times the number of non-controlled empty points, of the diagonal neighbors.

We can classify all space and prisoner points of a group according to Tables 1 and 2.

4. Perfect pure eye-regions

A set of adjacent eye-point is referred to as an eye-region. An eye-region consisting of only empty eye-points without any prisoners is called a pure eye-region. We shall investigate the number of eyes that a pure eye-region can provide in Sections 4–6. And we shall study eye-regions containing prisoners in Sections 8–10. Combining multiple eye-regions will be discussed in Section 7.

If a pure eye-region composes of only full eye-points without other adjacent empty point, and the surrounding boundary does not have defect, it is referred to as a perfect pure eye-region. Obviously, a perfect pure eye-region with one or two eye-points can produce exactly 1 eye.

The following eye-regions have their customary names:

- A perfect eye-region with 3 eye-points, referred to as Straight-Three or

Curved-Three corresponding to its straight or curved shape, has 1.5 eyes.

- A perfect eye-region with 4 eye-points in a square shape, referred to as Square-Four, has only 1 eye.
- A perfect eye-region with 4 eye-points in a T-shape, referred to as T-Four, has 1.5 eyes.
- A perfect eye-region with 4 eye-points in a straight or curved line, referred to as Straight-Four or Curved-Four respectively, has 2 eyes.
- A perfect eye-region with 5 eye-points in a shape like '+', referred to as Flower-Five, has 1.5 eyes.
- A perfect eye-region with 5 eye-points in which 4 in a square shape, referred to as Knife-Five, has 1.5 eyes.
- A perfect eye-region with 6 eye-points in which 4 in a square shape and other two adjacent to one corner of the square, referred to as Flower-Six or Grape-Six, has 1.5 eyes.
- A perfect eye-region with 6 eye-points in a rectangular shape, referred to as Rectangular-Six, has 2 eyes.

When we say an eye-region has 2 eyes, we mean it can produce 2 eyes regardless how the opponent attacks. Similarly, When we say an eye-region has 1.5 eyes, we really mean that the region produces 2 eyes if the group side plays first and 1 eye when the opponent moves first. Using the notation of combinatorial game theory, we can indicate the eye number situation precisely in this case as $\{2|1\}$. The number to the left (resp. right) of vertical bar indicates the number of eyes the region has when the group side (resp. the opponent) plays first. Thus, T-Four has $\{2|1\}$ eyes. Often we simply use the average and say T-Four has 1.5 eyes.

A group containing more than 2 eyes is the same as 2 eyes, being alive, so we will regard more than 2 eyes as 2 eyes. A perfect eye-region with 5 or 6 eye-points in a shape other than Flower-Five, Knife-Five, or Flower-Six has 2 eyes. Any perfect eye-region with 7 or more eye-points has 2 eyes.

A group containing just 1 eye-region with 1 eye or less and no other favorable factors (half eye-points, exit, or defect of surrounding opponent) is dead. A group containing just 1 eye-region with 1.5 eyes and no other favorable factors is critical.

Let us define external boundary of a region R, Ext(R) following Benson's notation, as follows:

Ext(R) is the set of grid and border points which are adjacent to some point in R but themselves are not in R.

The number of eyes that a perfect eye-region can produce, can be determined by the length of its external boundary according to the Table 3.

For example, Knife-Five has the length of external boundary 9 and it contains a Square four, so it provides 1.5 eyes. The readers are encouraged to check other common shapes of eye-regions. The idea of using length of Ext(R) to estimate the number of eyes was originally proposed by Mark Boon (personal communication).

Table 3
Number of eyes in a perfect pure eye-region

Length of Ext(R)	No. of eyes
$\leqslant 6$	1
7	1.5
8 (Square Four)	1
8 (Curved Four)	2
8 (Any other shape)	1.5
9 (Containing a Square Four)	1.5
9 (Not containing a Square Four)	2
$\geqslant 10$	2

5. Imperfect pure eye-regions

An eye-point adjacent to a false empty eye-point should be downgraded by one level i.e. a full eye-point is downgraded to a partial eye-point and a partial eye-point is downgraded to a false eye-point. If the opponent plays at the false eye-point neighbor, the eye-point will become a non-eye-point. If the group side plays at the false eye-point neighbor, the original status of the eye-point will regain. So a full eye-point adjacent to a false eye-point plays a similar role as a half eye-point not adjacent to a false eye-point.

The number of eyes that a region provides can be determined by the following rule.

Estimating rule for space eye-regions: After downgrading rules are applied,

Case 1: The region has only false eye-points left: it provides 0 eyes.

Case 2: The region has partial eye-points but no full eye points: it provides 1/2 eyes.

Case 3: The region has contiguous full eye-points left: we neglect partial eye-points to get a condensed pure eye region. Then use Table 3 on the condensed region to get the number of eyes. When a half eye-point is adjacent to a square four and with two or less empty neighbors, we add 1/2 to the result. If this number exceeds the one for the region counting both full and partial eye-points according to Table 3, then use the number for the combined region.

Case 4: The region has noncontiguous full eye-points: look-ahead search is needed in this case. We can use the region counting both full and partial eye-points to access Table 3 for an upper bound on the number of eyes.

The eye-region in Fig. 8 has 5 full eye-points, two of which marked by P are adjacent to false eye-points marked by A. These two Ps should be downgraded to partial eye-points. Now the condensed eye-region conforms to a Straight-Three after neglecting points Ps, so it has 1.5 eyes. When one (or both) of the white stones on line 1 is absent, the adjacent A is opponent-accessible, the number of eyes is the same as that in the figure.

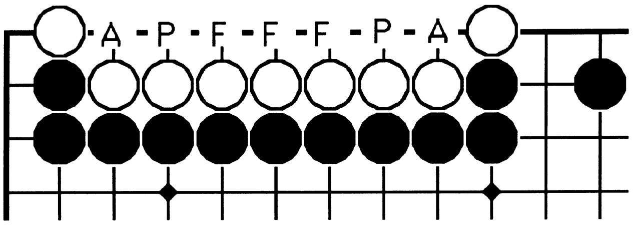

Fig. 8.

In Fig. 9, neglecting the *E*, which is downgraded to a partial eye-point for being adjacent to a false eye-point, the eye-region conforms to Knife-Five, so it has 1.5 eyes. Note that if the upper left *F* is occupied by a white stone, then the region with point *E* neglected is of 2-eye shape but the original region is a Knife-Five with 1.5 eyes. We should take the smaller number – the region has 1.5 eyes.

In Fig. 10, neglecting *E*, the eye-region is square four, so the basic rule predicts 1 eye. However, *E* is adjacent to a Square-Four of full eye-point and it has no more than two empty neighbors, so it has 1.5 eyes. White can rescue the group by moving at the right adjacent of *E*. A group conforming to square four

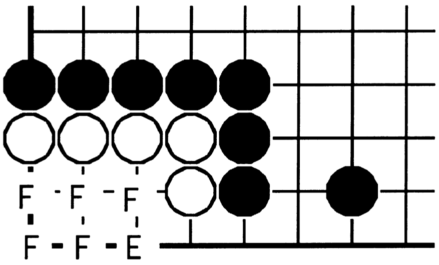

Fig. 9.

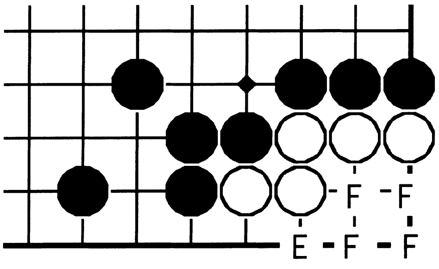

Fig. 10.

with the neglecting of a full eye-point that is adjacent to a false eye-point can generally be rescued.

The white group in Fig. 11 conforms to knife five after neglecting the downgraded point *E*, but *E* is adjacent to a Square-Four. So the region has 1.5 + 0.5 eyes. It is already alive.

Let's re-state our first downgrading rule on eye-points below.

Downgrading rule 1: For each false empty eye-point, the adjacent empty eye-point should be downgraded by one level.

The following are some additional examples.

In Fig. 12, *A* is a false eye-point, so we should downgrade *B*. Resulting condensed eye-region has two adjacent full eye-points. It has only 1 eye.

If the black stone diagonal to *A* were not there. The condensed eye-region forms a Straight-Three with 1.5 eyes.

In Fig. 13, after downgrading *B*, the eye-region contains only one partial eye-point, so it has 0.5 eyes.

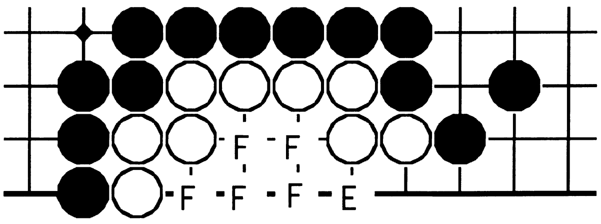

Fig. 11.

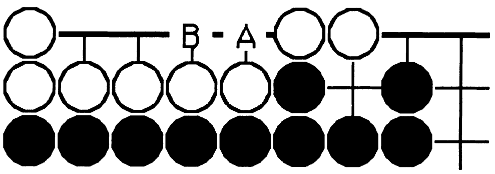

Fig. 12.

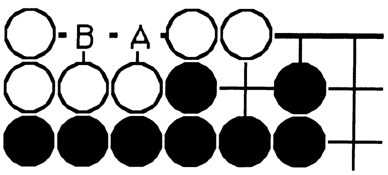

Fig. 13.

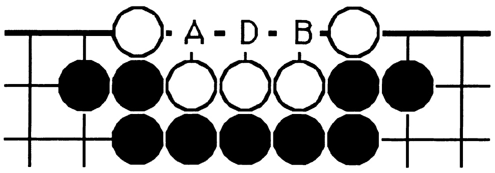

Fig. 14.

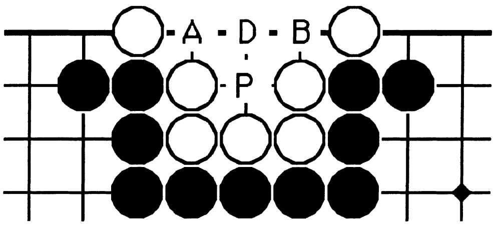

Fig. 15.

In Fig. 14, since *A* is a false eye-point, we downgrade *D* to a partial eye-point. Now *B* is also a false eye-point, so we further downgrade *D* to a false eye-point. Hence the region has no eyes.

In Fig. 15, again here *D* is downgraded to a false eye, so we should downgrade its neighbor *P* to partial eye-point. Hence the region has 0.5 eyes.

If neglecting a point breaks the eye-region into two parts, we should not neglect the point. In this case, if the region without the neglecting has $n < 2$ eyes, then we use the *n* as the number of eyes; otherwise a look-ahead search is needed for determining the number of eyes.

6. Combinatorial eye values

Numeric eye numbers are sometimes inappropriate.

In Fig. 16, if we play first at *C*, it produces 2 eyes. If the opponent moves first at *C*, it has no eyes. Thus it has $\{2|0\}$ eyes. In this case it is rather dangerous to use the average value 1. Since this eye-region plus one ordinary eye do not guarantee 2 eyes. The region in Fig. 17 has $\{\{2|1\}|0\}$ eyes. The white block has no eye if Black moves at *C*, while it becomes the shape similar to Knife-Five if White moves at *A*, getting 1.5 eyes. Landman [7] contains many example eye-regions with complicated combinatorial game theoretic eye number values. We shall focus on simple common eye-regions frequently occurred in actual games, so a set of generally applicable heuristic rules for life/death recognition can be developed.

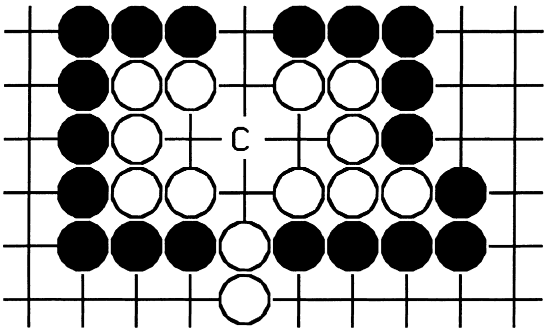

Fig. 16.

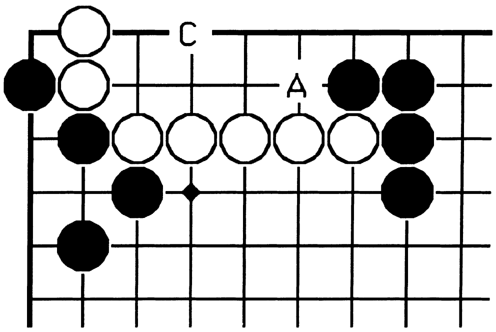

Fig. 17.

7. A group with multiple eye-regions

When a group has several mutually independent eye-regions, the number of eyes individual regions have can be combined to determine the number of eyes that the group has. The eye number of a region can be presented as a binary tree with nonnegative integers attached to the leaves as our notation in Section 6 suggests. A crisp integer number of eyes can be viewed as a degenerated binary tree with just one leaf (the root) attached with the integer value. Kao [5] proposed effective numerical algorithms for finding the means and temperatures of such binary game trees. For our eye number evaluations, some simple computational methods will suffice, which we shall discuss in this section.

The exact number of eyes can be calculated through a simple combinatorial search on the binary trees as follows: total number of eyes is initialized to the sum of values of the degenerated trees. The group side gets to select a non-degenerated tree and to replace it with the left subtree. The opponent gets to select a non-degenerated tree and replace it with its right subtree. When a tree becomes a single leaf node, the attached number is added to the total number of eyes. The group side wants to maximize the total number of eyes and the opponent wants to minimize it. This mini–max search is much simpler than regular life and death search of the group with grid points as options.

When all the binary trees involved, one for each eye-region, are integers *ai* or switches $\{bj|cj\}$ where *ai*, *bj*, *cj* are all non-negative integers. Combinatorial game theory, see [3], tells us to play at the biggest switch, i.e. the one with biggest *bj–cj*. As a consequence, the number of eyes that a group has can be easily calculated without search:

Proposition 1. *Assume a group has multiple mutually independent eye-regions with eyes* $a1, a2, \dots, an, \{b1|c1\}, \{b2|c2\}, \dots, \{bm|cm\}$, *such that* $b1 - c1 \geqslant b2 - c2 \geqslant \cdots \geqslant bm - cm$. *Then the group has* $\{p|q\}$ *eyes where* $p = a1 + a2 + \cdots + an + b1 + c2 + b3 + c4 + \dots$ and $q = a1 + a2 + \cdots + an + c1 + b2 + c3 + b4 + \dots$ eyes.

If $p < 2$ then the group is dead.
If $q \geqslant 2$ then the group is alive.
If $p \geqslant$ and $q < 2$ then the group is critical.

Proposition 2. *If all the switches in Proposition* 1 *are of size* 1 (*i.e.* $bi - ci = 1$). *Then we can simply add average number of eyes of each region to get total number of eyes* $= a1 + a2 + \cdots + an + 1/2(b1 + c1) + 1/2(b2 + c2) + \cdots + 1/2(bm + cm)$.

So if each eye-region can provide a whole number or a whole number plus a half eyes, then we can simply add all eye numbers in an ordinary way. If the sum is $\geqslant 2$, the group is alive. If the sum is $\leqslant 1$, it is dead. If the sum is 1.5, the group is critical.

In Fig. 18, the two eye-region of the white group are mutually independent. One is a curve-three with 1.5 eyes; the other has 0.5 eye. Total number of eyes = 1.5 + 0.5 = 2. The group is alive. Mutual independence of the eye-regions is an important precondition for summing up eye numbers. For example, Fig. 19. The two eye-regions of the white group are not mutually independent. Black's move at *A* can threat both eyes. We can not just add 1 + 1 and claim it has 2 eyes.

8. Semi-pure eye-region

An eye-region containing both empty eye-points and prisoner eye-points is said to be semi-pure if all prisoners are single stones with one liberty.

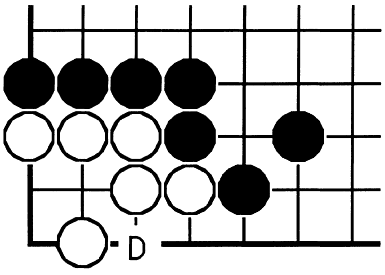

Fig. 18.

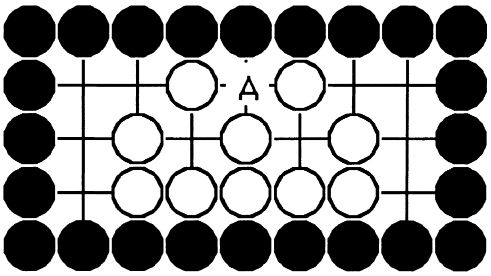

Fig. 19.

The eye-regions in Figs. 20–22 are all semi-pure. Pure eye-regions and semi-pure eye-regions are collectively called space eye-regions.

The following is a set of heuristic rules to deal with prisoners and their surrounding points in semi-pure eye-region:

Downgrading rule 2: If the one-liberty prisoner is a full eye-point, then its liberty is immune from any downgrading rules.

Downgrading rule 3: If the prisoner is a false eye-point, doubly downgrade its liberty to false eye-point (and then, applying rule 1, downgrade this liberty's empty neighbors by one level.

A downgraded eye-point should not be neglected, if it may increase the eye number estimation or it may break the region into two or more parts.

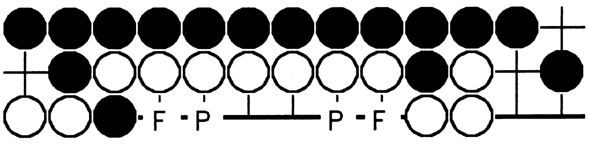

Fig. 20.

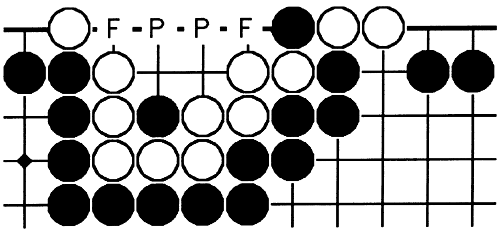

Fig. 21.

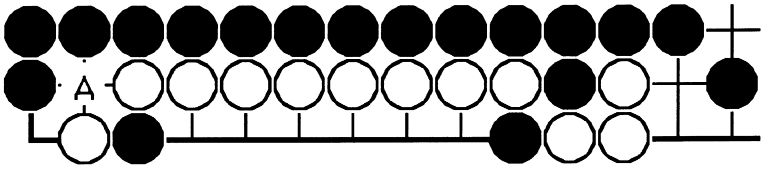

Fig. 22.

After the neglecting non-full-eye points as required by downgrading rules, we can treat the resulting region as a perfect pure eye-region and use Table 3 to determine the number of eyes.

For example, after applying downgrading rules, the group in Fig. 20 has two full eye-points side by side, so it has 1 eye. The group in Fig. 21 has a Curved-Three full eye-point region, so it has 1.5 eyes.

Adjustment: When the region is a corridor and two ends of the corridor are both prisoner false eye-points, we should adjust the eye evaluation by adding 1/2 eyes unless there are no eye-points (full or partial) left after downgrading. For example, Fig. 23, Case 2 of Eye number estimating rule in Section 5 says the group in Fig. 23 has 1/2 eyes. Since both ends of this linear region are prisoner false eye-points, we should adjust the answer by adding 1/2 eyes. This region really provides 1 eye.

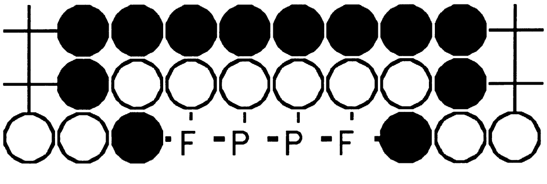

Fig. 23.

If there is a prisoner half eye-point, such as the left end of Fig. 22, we can estimate the number of eyes as {NL|NR} where NL (NR) is number of eyes of the region with the empty point A replaced by a white stone (black stone).

9. Prisoner eye-region

Dead opponent stones may provide eye-points for a group. Dead opponent stones may be full, half, or false eye-point just the same as empty points. Opponent dead stones can form eye-region by themselves or together with empty eye-points. We note that the liberties of opponent's dead stones make little contribution to the number of eyes. To convert opponent's dead stones into eyes, we have to take them off the board. The adjacent empty points (liberties) are thus filled with our stones and the group makes no eye at those points. If an eye-region consists of one block of prisoners (dead opponent stones) and its liberties only, then the eye-region is called a prisoner eye-region. We shall investigate the number of eyes in prisoner eye-regions in this Section.

Estimating rule for Prisoner eye-region: we can neglect prisoner false eye-point and its immediate neighboring prisoner. We can downgrade prisoners adjacent to an empty false eye-point by one level.

Case 1: The dead opponent block has no more than one full eye-point liberty. If the full prisoner eye-points of the dead block has the shape of a pure eye-region with $\leqslant 1.5$ eyes, the region has 1 eye. If it is of a shape worth 2 eyes, then the region has 2 eyes. For example, Figs. 24 and 25.

The prisoner eye-region in Fig. 24 only has one full eye-point liberty and the dead block is of Knife-Five shape, so the white group has only 1 eye.

In Fig. 25, the left most two prisoners should be ignored. We have a prisoner eye-region of shape Straight-Three. So it has 1 eye.

Case 2: The dead opponent block has two full eye-point liberties and no other liberties.

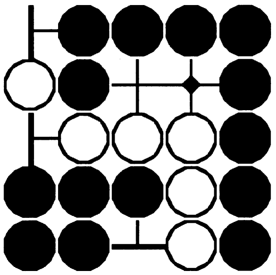

Fig. 24.

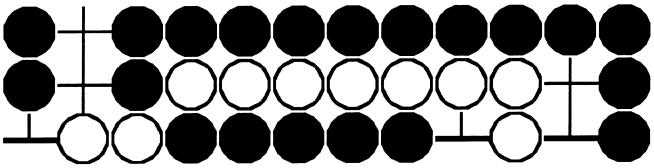
Fig. 25.

Subcase 2.1: The full prisoner eye-points of the dead block form the shape of a pure eye-region with 2 eyes.

If both Black expansion at liberty are of 2-eye shape, then the region has 2 eyes otherwise it has 1.5 eyes.

Subcase 2.2: The full prisoner eye-points of the dead block form the shape of a pure eye-region with <2 eyes.

If White has no other eye-regions then Black's 'dead' block is actually alive (either seki or the white group is dead). If at least one expansion is <2 eye shape then the region has only 1 eye, otherwise, it is seki. For example, Figs. 26 and 27.

In Fig. 26, the dead black block is of 2-eye shape but there is one expansion of the dead black block of <2 eye-shape. The region has 1.5 eyes and the white group is critical.

In Fig. 27, the Straight Three is of <2-eye shape and either expansion of the black dead block is of 2-eye shape (Straight-Four). So this is a seki, both white and black are alive.

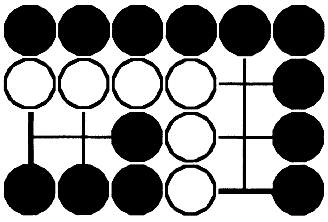
Fig. 26.

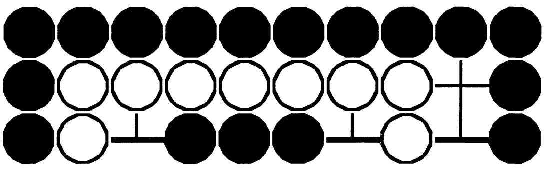
Fig. 27.

Case 3: The dead opponent block has $n > 2$ full eye-point liberties. This can be settled recursively in terms of dead block with $n - 1$ full eye-point liberties as follows. We note that an eye-region is reducible to under 2 eyes if and only if there is a one stone black expansion that can make the resulting region worth $\leqslant 1$ eye according to rules in Case 2 and that an eye-region is fixable for 2 eyes if and only if White can play at one of the dead black block's liberties to make the resulting region worth 2 eyes according to rules in Case 2.

Thus our rule for Case 3: if the eye-region is reducible to less than 2 eyes then if the eye-region is fixable for 2 eyes then it has 1.5 eyes else it has 1 eye else it has 2 eyes. For example, Fig. 28. Since the eye-region is both reducible to under 2 eyes and fixable for 2 eyes (the key point is the empty point in the middle), the white group in Fig. 28 has 1.5 eye.

Case 4: The region has a prisoner half eye-point.

We will do the estimate as $x|y$, where x is the number of eyes of the region with the half eye-point treated as a full eye-point and y is the number of eyes of the region with the half eye-point treated as a false eye-point. For example, Fig. 29 , the white group in Fig. 29 has 2|1, or 1.5, eyes.

Case 5: The region has two half eye-points.

We count one half eye-point as a full eye-point and the other as a false eye-point to get eye estimate $e1$ then reverse the roles of the two half eye-points to get eye estimate $e2$. The region has $d1|d2$ eyes where $d1$ is the max(ceiling of $e1$, ceiling of e2), $d2$ is the min(floor of $e1$, floor of $e2$), where ceiling of e is the smallest integer which is $\geqslant e$, and floor of e is the largest integer which is $\leqslant e$.

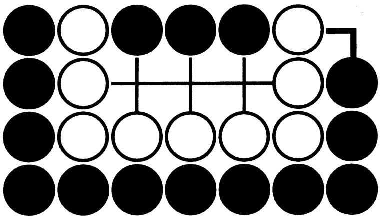
Fig. 28.

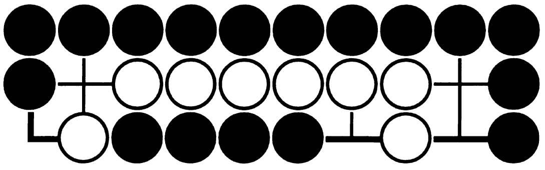
Fig. 29.

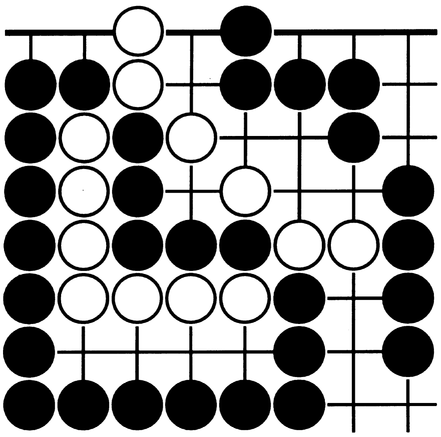
Fig. 30.

For example, Fig. 30. After we ignore one half eye-point prisoner and its neighbor prisoner in Fig. 30, the black dead block becomes Straight Three shape; so it has only 1 eye.

10. Mixed region

If an eye-region, consisting of empty points and dead opponent stones, does not conform to any of the types discussed in Sections 4–9, it is called a mixed region. Additional downgrading rules will be needed for prisoners.

Downgrading rule 4: For each false empty eye-point that is not forbidden to the opponent, the adjacent prisoner eye-point should be downgraded by one level.

Downgrading rule 5: For each prisoner, the empty neighbors should be downgraded to non-eye-points.

Downgrading rule 6: For each false prisoner eye-point, the adjacent prisoner eye-point should be downgraded to non-eye-point.

We shall call an empty point in the region a separating point for the region if after White plays at the point, the region becomes two separate eye-regions with total number of eyes $\geqslant 2$.

Eye estimating rule for mixed regions:

Case 1: If the region has two or more non-opponent-sente separating points, or has one separating point that is a forbidden point for opponent, it has 2 eyes.

The group side can get to play at least one separating point and make 2 eyes. If the opponent gets to play the separating point as sente, then we can not rely on that separating point.

Case 2: The region has one non-forbidden separating point. The number of eyes is $2|x$. x can be determined as follows: let Black place a stone at the separating point. If the resulting region has at least 1.5 eye, then $x = 2$ else $x = 1$.

11. Implementation

The rules mentioned in this paper are fully implemented in Go Intellect. The first author has tested the implementation with the 60 problems in the book (Chinese translation version) 'Basic Life & Death in Go' edited by Nihon Kiin. On the 20 questions in Chapter 2, which tests human players' intuition, GI gets 17 of them correct using the static analysis. On the 30 questions in Chapter 3, which requires look-ahead, GI gets 21 of them correct by doing only the static analysis. On the more open problems in Chapter 4, GI gets only 3 out of 10 without performing look-ahead. Overall, the static analysis scores 41 out 60 and is ranked as 5th kyu by the book. HandTalk implements a variation of the static rules in the paper and performs very well on life/death judgements at computer Go tournaments.

Life and death is a fundamental problem in the game of Go and is intrinsically highly complex. In order to conquer combinatorial explosion, we believe the best approach for a Go program to handle the life/death problems is as follows:

Let static analysis handle those cases covered by the heuristic rules (90% or so), and leave the other 10% to heuristic search using static analysis as a base to determine and to evaluate terminal nodes in the search tree. Full-blown thorough searches take place only during opponent's time as bonuses.

References

[1] D.B. Benson, Life in the game of Go, Information Sciences 10 (1976) 17–29.
[2] D.B. Benson, A mathematical analysis of Go, in: K. Heine (Ed.), Proceedings of the Second Seminar on Scientific Go Theory, Institut fuer Strahlenchemie, Muehlheim a.d. Ruhr, 1979, pp. 55–64.
[3] E.R. Berlekamp, J.H. Conway, R.K. Guy, Winning Ways, Academic Press, New York, 1982.
[4] K. Chen, Group identification in computer Go, in: D. Levy, D. Beal (Eds.), Heuristic Programming in Artificial Intelligence, Ellis Horwood, Chichester, 1989, pp. 195–210.
[5] K. Kao, Sums of hot and tepid combinatorial games, Ph.D. Thesis, University of North Carolina at Charlotte, 1997.
[6] A. Kierulf, K. Chen, J. Nievergelt, Smart game board and Go explorer: a study in software and knowledge engineering, Comm. ACM 33 (2) (1990) 152–166.
[7] H. Landman, Eyespace values in Go, in: R. Nowakoski (Ed.), Games of No Chance, MSRI Publications 29, 1996, pp. 227–257.

[8] D. Lichtenstein, M. Sipser, Go is polynomial-space hard, Journal of ACM 27 (2) (1980) 393–401.
[9] M. Mueller, Computer Go as a sum of local games: an application of combinatorial game theory, Ph.D. dissertation, Swiss Federal Institute of Technology, Zurich, 1995, pp. 62–64.
[10] R. Popma, V. Allis, Life and death refined, in: J. Herik, V. Allis (Eds.), Heuristic Programming in Artificial Intelligence 3, Ellis Horwood, Chichester, 1992, pp. 157–164.
[11] J.M. Robson, The complexity of Go, in: Proceedings of the IFIP, 1983, pp. 413–417.
[12] T. Wolf, Investigating Tsumego problems with "RisiKo", in: D. Levy, D. Beal (Eds.), Heuristic Programming in Artificial Intelligence 2, Ellis Horwood, Chichester, 1991, pp. 153–160.