A model of visual organization for the game of GO

by ALBERT L. ZOBRIST

University of Wisconsin
Madison, Wisconsin

INTRODUCTION

No successful GO-playing program has appeared in the literature, although Remus[1] used GO as the subject of a machine learning study, and Thorp and Walden[2] have considered some of its mathematical aspects. Another author[3] considered GO to be somewhat mysterious, making it a challenge to those interested in automating it. Apparently the game was described as being mysterious to indicate that people were able to play it without knowing how they were able to play so well. More study of this complex game may reward us with new insight into human perceptual and problem solving abilities as well as foster the development of new techniques for artificial intelligence. This report describes a program which plays GO. The program uses an information processing model to produce perceptual features which are seen by human GO players, and is capable of several responses to the recognition of significant configurations of these perceptual features.

A brief description of GO

The rules of GO are deceptively simple. The two players alternate in placing black and white stones on the intersections of a 19 × 19 grid. Stones of the same color which are connected by row or column adjacency form a *chain*. Diagonal adjacency is not sufficient to connect a chain. The empty intersections which are adjacent to a chain are its *breathing spaces*. When a chain has no breathing spaces, it is captured by the opponent, and the captured men are removed from the board. A player may place his stones anywhere on the board with two exceptions: (1) he may not form a chain with no breathing spaces unless he is capturing, and (2) he may not capture one stone which has just captured one of his stones on the previous turn. A player may choose to pass at any turn. The game is over when both of the players pass in sequence. A player's score is the sum of territories surrounded by his color plus the number of opponent's stones captured.

Some of the basic consequences of these rules are illustrated by the right side of Figure 1. White can always capture the top black chain, but cannot capture the bottom black chain, if black moves Figure 1 properly. If black moves at either of T2 or T3 then white cannot occupy all of the breathing spaces of the black army without committing self-capture. This is because the black army would have two separate *eyes*. The ability to form two eyes is what determines whether an army is safe or not. White will score 16 points in the upper right, and black will score four points in the lower right corner. If white moves R10, then black may not respond R11, but must move elsewhere on the next turn. This prevents cyclic capture.

The rules scarcely describe how GO is actually played. Interested readers are advised to seek a demonstration from someone who plays GO, or to read one of the beginner's books.[4,5] The situations in the left hand corners of Figure 1 are representative of real play. Although the stones are not connected into long chains, they threaten to form chains which will surround territory along the corners and edges of the board. Efficient play requires that as much territory be sketched out with as few stones as possible. Throughout the rest of this paper such aggregates of stones which threaten to invade or surround territory will be called *armies*.

The problem of complexity

GO is considered more difficult than chess by many people who know both games.[5] Numerical measures of the complexity of checkers, chess, and GO tend to support this belief. The number of paths down the move

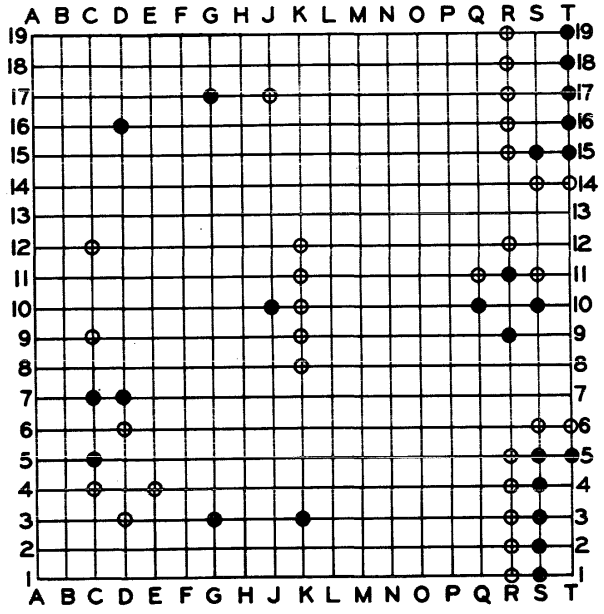


Figure 1—An illustration of GO

tree has been estimated at 10^{40} for checkers[6] and 10^{120} for chess.[7] A rough estimate for the number of paths down the move tree for GO is 361! or 10^{761}. By this reasoning, GO played on a 6 × 6 board would be comparable to checkers in complexity, and GO on a 9 × 9 board would be comparable to chess.

A slightly better extimate of the true complexity of these games may be obtained. For checkers, suppose that a choice of three reasonable moves occurs approximately 20 times per game. Then 3^{20} is a crude estimate of the number of checker games which might occur in ordinary play. Good chess players usually consider less than five move choices, hence 5^{50} estimates the number of reasonables chess games. A typical GO game lasts about 300 moves and a choice of 10 reasonable moves occurs at least 100 times, thus there are at least 10^{100} GO games which could occur in humn play.

Such calculations, however crude they may be, are important to anyone interested in the automation of these games. The complexity of GO may hinder attempts to program it with the methods developed for chess and checkers.[6,7,8,9] The move tree for GO is exceedingly deep and bushy, hence any form of heuristic search can explore only a relatively small portion of the complete tree. An alternative approach might be to concentrate upon extremely powerful methods of evaluation of the board situation, thus enabling better play with a more restricted search. Another possibility might be to have the lookahead be directed by pruning methods which correspond to the development of strategies. Time will tell whether a successful GO playing program can be written using such methods.

The visual nature of GO

The recognition and discrimination of meaningful perceptual stimuli presupposes the active formation of stable perceptual elements to be recognized and discriminated. A person lacking this process would combine all sorts of stimuli into meaningless groups.[10] The choice of a move in GO usually involves the recognition of configurations which are meaningful to the player. This raises the question as to whether certain basic perceptual processes are necessary for the comprehension of a GO board. The following examples might suggest that the answer is yes.

First, consider the spontaneous grouping of stones of the same color which occurs during visualization of a GO board. The stones are organized into distinct groups, clusters, or armies even though they may be sparsely scattered about or somewhat intermingled. Grouping is usually the result of proximity of stones of the same color or the predominance of stones of one color in an area, but can be affected by other characteristics of the total board situation. For example, stones which fall into a line are likely to be grouped. Kohler[11] and others have found grouping to be a basic perceptual phenomenon. Yet the recognition and discrimination of groups or armies is necessary for competent GO play.

Closely related in grouping is segmentation, which is also discussed in Kohler. The area subtended by the board is divided into black and white territories, each of which maintains its own integrity in the visual field. These segments are a measure of the territory which is controlled by either side, hence are an important factor in the assessment of a GO board.

Another example is the formation of "spheres of influence" about a stone or group of stones. Influence is not an inherent property of stones, but appears to be induced in them by our processess of perception. Yet they are a crude measure of the potential of a stone or army of stones for controlling territory on the board.

The spontaneous image formed by the visualization of a GO board appears to be a complicated assemblage of perceptual units and subunits. For example, the stones themselves have their own perceptual identity while at the same time they are parts of chains or groups of stones. The phenomena discussed above show that some of these perceptual processes may be very important to the ability of GO players to comprehend this complex game.

It is not within the scope of this report to discuss further the psychological nature of these perceptual

mechanisms, or to speculate upon the physiological basis for them. Let us adopt the term *visual organization* to mean the formation of such stable perceptual elements as have just been discussed, and let the resulting "mental picture" be called the *internal representation.*

Given that a player "sees" a fairly stable and uniform internal representation, it follows that familiar and meaningful configurations may be recognized in terms of it. The result of visual organization is to classify a tremendous number of possible board situations into a much smaller number of recognizable or familiar board situations. Thus a player can respond to a board position he has never encountered, because it has been mapped into a familiar internal representation.

This report will describe a simulation model for visual organization. It will use transformations which create information corresponding to the perceptual features discussed above, storing them in a computer internal representation.

A heuristic for visual organization

We now examine the problem of modeling the basic visual organization of the GO board. A reasonable goal would be to determine the segmentation of the board, the domains of influence of the stones, and the armies of stones, storing that information in a computer internal representation. Before building the computer model, it is of interest to consider physical processes which give some measure of the influence of physical bodies.

There are many candidates in the physical sciences for the process we desire. For example, white stones could be electrons, and black stones could be protons. Contiguous areas of positive or negative potential could determine the segmentation of the board, and the value of the potential would measure the influence of the stones. Of course, the solution would be discretized to the points of the GO board, and the potential at an unoccupied point could determine how well protected that point is by black or by white.

For another candidate, let the GO board be made of blotter paper and simultaneously place a drop of oil under each black stone and a drop of water under each white stone. Contiguous areas of oil or water would determine the armies and the segmentation of the board. Since the oil and water would spread evenly, the concentration would not indicate the influence of the stones or even their location.

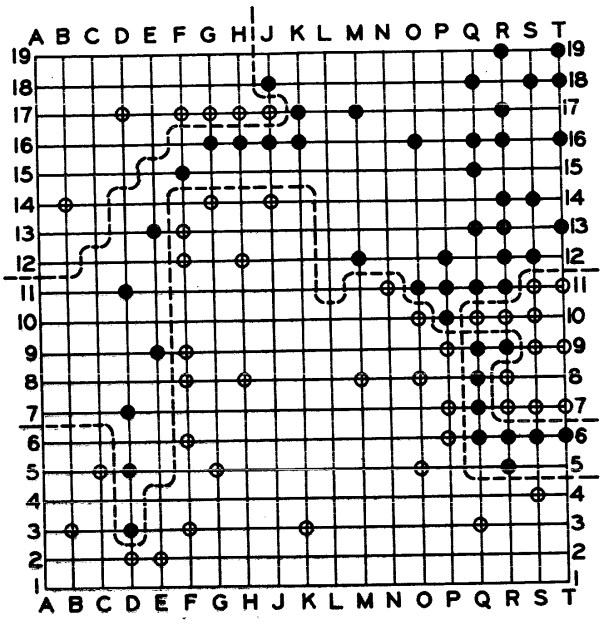
Other possibilities might involve electrical networks or heat conduction, etc. These physical models are considered because they are well defined and easily calculated, whereas the visual process we are attempting to model is ill defined. Let us consider in more detail the first two examples given above. In the center of Figure 1, the electric charge analogy would give the black stone at J10 some weight to the right of the wall of white stones, whereas oil from the black stone would never get past the wall of water spreading from the white stones. Perceptually speaking, the black stone has no influence to the right of the white stones. The oil and water analogy could not differentiate between the two situations in the right hand corners of Figure 1, whereas the electric charge analogy would show four strongly surrounded squares in the lower corner. Thus the oil and water analogy does not reflect our perception of this situation. The finite difference method used by the program was chosen with both of these models in mind, and has the good features of both.

It is assumed that a game is in progress and the board position is stored. The position is transferred to a 19×19 integer matrix by placing 50 for each black stone, -50 for each white stone, and 0 elsewhere. Then each point which is positive sends out a $+1$ to each of its four neighors, and each negative point sends out a -1 to each of its neighbors. These numbers are accumulated as this procedure is repeated four times. Figure 2, which is taken from the game listed at the end of this report, illustrates the results of the visual organization heuristic. The negative integers are indicated by an underline.

Segmentation can be assessed by determining the contiguous areas of positive or negative integers. The dashed lines in Figure 2 indicate the resulting segments. The stones which lie in a segment may be considered to be a group or army. The integer values at a point give a measure of the degree of influence exerted by the stones nearby. The influence of stones of the same color may reinforce one another, whereas the fields from opposing stones seem to repel and cancel one another. Inspection of Figure 2 should convince us that at least a crude resemblance to perceptual processes has been obtained. The array of integers from the visual organization heuristic contains, at least implicitly, information corresponding to an internal representation. This heuristic, together with a routine which is capable of creating an explicit computer internal representation of the resulting information, will be part of a model of visual organization. The specific details of the entire GO-playing program will now be given.

The program

The program is written in ALGOL for the Burroughs B5500 computer. Interaction is provided by remote teletypes. Each move requires 5 to 8 seconds of central




```
0  2  4  5  6  6  4  1  7  7  6  5  5  5  7 10 59 12 57
2  4  8 10 10 11 11  2 50 12 10 10  9  9 10 62 16 63 61
3  7 10 62 10 57 57 56 42 56 13 62 12 11 12 14 63 14 11
5  8 10  6  0  4 56 57 56 64 12 12 12 62 13 64 64 14 59
7 10  8  0  7 56  7  6  6  5  8  9  9 11 12 63 15 13 10
8 62  6  3  6  1 56  8 57  3  3  6  8  8 11 14 64 63 11
7  9  1  7 54 56 14 13 12  5  4 10  8 10 12 63 65 16 59
2  0  3 11  6 58 13 62 10  2  7 58  5 12 63 16 65 56  4
1  4 10 62  6  6 11 10  7  1  2  0 47 49 66 57 50 50 54
2  5  9 12  7  6 10  9  6  3  2  7 12 48 42 42 50 65 12
1  4  8 12 54 56 12 11  8  6  8 10 12 14 48 50 42 57 60
2  5  9 11  5 58 13 62 10  8 10 62 12 62  8 51 49 15 11
1  3  7 61  4  8 12 10  8  7  8 10 11 13 56 50 50 57 53
3  3  0  8  3 58 12 10  7  5  6  8 10 13 56 57 58 57 53
6 11 53 54  1  9 62 10  8  7  7  7 10 62  7  2 58  7  4
8 12  6  4  1 11 12 10 10 10  8  8  8 10 12  5  6 55  3
8 61  6 44  5 62 11  9 10 62 10  6  6  8 10 62 11 11  7
7 11 11 56 63 12  8  6  8 10  8  4  3  4  8 10  9  7  4
4  6  8  9  9  7  4  3  4  5  4  2  0  2  4  5  5  3  2
```

Figure 2—Results of the visual organization heuristic

processor time or about 5 to 20 seconds of real time, depending upon the number of users being serviced by the system. The machine code occupies 6300 words of core memory and 5400 more words are required for storage arrays. The program has two distinct parts. Part I has a coordinated set of procedures, including the visual organization heuristic, which operate on the board position to produce a computer internal representation. Part II has a set of procedures which use the internal representation to calculate a move.

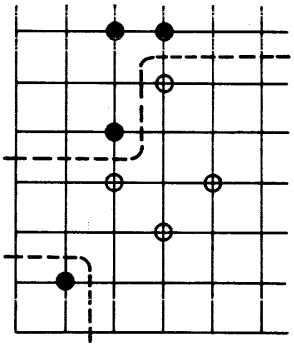
Part I realizes a model of visual organization for GO, producing an analogue of a human player's perception of the board. Part II is not an attempt at simulation, but a collection of heuristics which may or may not resemble cognitive processes.

Part I

This part of the program consists of a set of computations which transform the board position into a computer internal representation. The internal representation contains an analog of important perceptual features of the board position. This information is stored in seven 19×19 integer arrays in an explicit fashion. That is, the integer values are a direct measure of the features they represent.

For example, consider the features of perceptual grouping and segmentation which have been determined by the visual organization heuristic. It would not be easy to reference this information in the array shown in Figure 2. Another process must create an array which gives a direct measure of the size of the segments and the groups of stones.

Figure 3 illustrates the results of the processes which



```
603 603 603 603 603 603
603 603 603 704 704 704
603 603 603 704 704 704
704 704 704 704 704 704
704 704 704 704 704 704
501 501 704 704 704 704
501 501 704 704 704 704
```

Figure 3—Internal representation of grouping and segmentation

calculate perceptual grouping and segmentation. These processes act upon the array of integers produced by the visual organization heuristic to produce an array of integers which are a part of the internal representation. One of these numbers (e.g., 603) may be interpreted as follows: the hundreds indicate the size of the segment which covers that point (600 is a medium-sized segment) and the ones indicate the number of stones which lie in that segment (there are 03 black stones hence 600 + 03 = 603). 500 indicates a small segment with less than 10 empty intersections, and 700 indicates a large segment with more than 25 empty intersections. If no segment covers an intersection, then 0 is stored in the corresponding cell of the array. The empty intersections in a segment are counted as they are a better measure of the safety of the army of stones in a segment. The information is compressed by having the size and the number of stones in the same array for purposes of efficiency only.

The array shown in Figure 3 is typical of the seven arrays which constitute the computer internal representation. Numerical values correspond in an explicit fashion to feature which we have considered to be formed by visual organization.

The second array gives a numerical measure of the influence of the black and white stones. It is an exact copy of the array of integers shown in Figure 2.

The third array measures the number of breathing spaces in a chain. This array is illustrated in Figure 4.

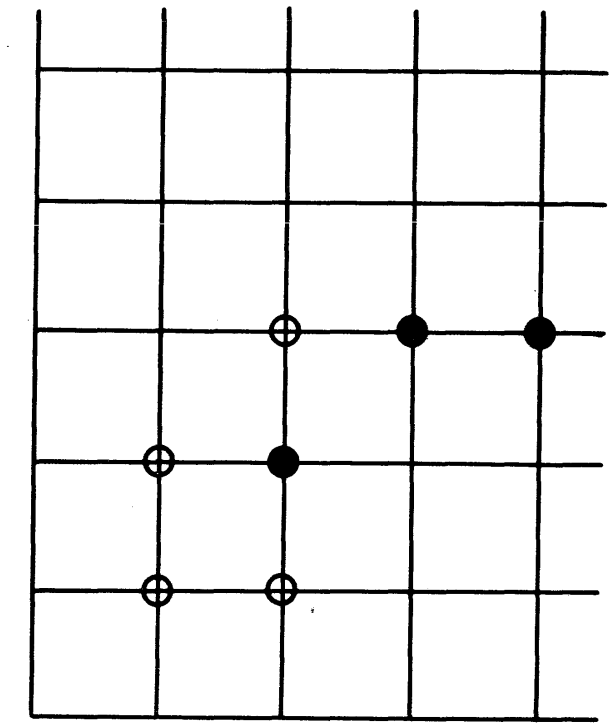
The fourth array measures the number of stones in a chain. This array is calculated in the same fashion as the third array.

The other three arrays of the internal representation contain integers which indicate the color of stones, the color of segments, and the number of stones of each color which are adjacent or diagonal to the points of the board.

Part II

The second part of the program uses the recognition of "familiar" configurations in the computer internal representation as the basis of its calculations. The mechanics of this recognition process will be described first.

Figure 5 illustrates a configuration which the program is able to recognize. At point A, there is a black stone which has only one breathing space left. Point B is an empty intersection. Point C has a black stone which may be part of a safe army of men. Note that the geometric arrangement of this 3-tuple is as important as the three features. The program contains a prototype of this configuration which we shall call a *template.*



0	0	0	0	0
0	0	0	0	0
0	0	2	5	5
0	6	1	0	0
0	6	6	0	0
0	0	0	0	0

Figure 4—Internal representation of breathing spaces

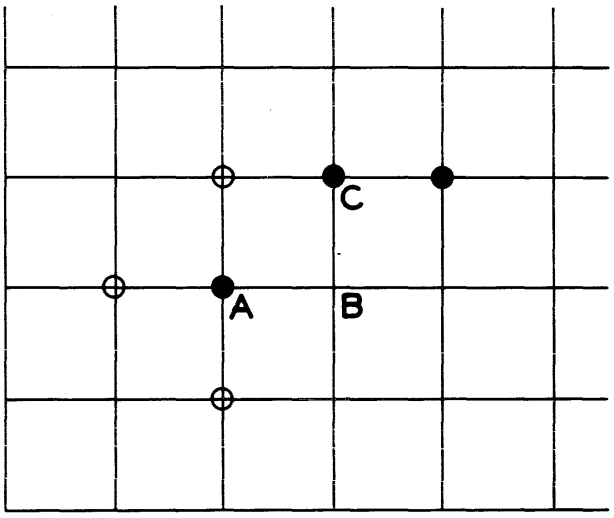


Figure 5—Illustration of a significant configuration

A template consists of an n-tuple of references to the internal representation together with a specification of the geometric arrangment of the elements of the n-tuple. Thus the template for our situation ABC is:

(0,0) black stone, 1 breathing space
(1,0) empty intersection
(1,1) safe black stone.

The pairs of numbers are the relative coordinates of the references. The references themselves must be translated into a numerical form which can be used to process the internal representation, for example:

safe black stone = 601 thru 900 in array 1 and
1 thru 1 in array 7.

That is, a point satisfies the reference "safe black stone" if the value of that point in array 1 of the internal representation lies between 601 and 900 inclusive and the value in array seven equals 1. A 1 in array seven tells us that we have a black stone on that point, and a 601 to 900 tells us that we have at least a medium-sized segment.

The program has the ability to scan such templates to all positions, all rotations, and all reflections of the board, thus recognizing the configuration ABC wherever it occurs. If the template is carefully specified, then configuration ABC can be quite a general occurrence. For example, Point A could be connected to a chain of stones with 1 breathing space left. Allowing this would give the template more generality.

The present program has 85 templates capable of recognizing a wide variety of configurations. A template that matches implies either a move or a heuristic lookahead procedure. In fact, there are two types of templates for these two purposes.

The first type of template specifies a weight of implication and a pair of relative coordinates for the move which is implied. For example, the template for configuration ABC described above would specify

(1,0) 500

which means that a weight of 500 is assigned to the point B in Figure 5. These weights are stored in a special 19 × 19 array reserved for this purpose. Several templates may imply the same move in which case the weights are summed in this array. The highest sum of weights indicated the best move. Bad moves and illegal moves usually have a negative weight.

One more example of this type of template will be given:

(0,0) white segment
(1,0) black segment
(0,0) weight 40.

This template implies a move with weight 40 at the interface between opposing segments. A weight of 40 is relatively small, hence it will merely give a tendency towards moving in these areas. This template is very general; it can match as many as 100 times in a single scan of the board. It gives a slight tendency to move between opposing armies which helps the program's play.

There are 65 templates which imply moves in the manner just described, the other 20 templates imply a heuristic lookahead. The difference between these two types of templates is that instead of a weight being placed in the weight array, an x is-placed in a 19 × 19 array as an indication of the template match. The x's are a mark to indicate that a move tree search should be performed in the local area about the x. All of the templates are applied before the search is begun. Figure 6 illustrates a configuration which would be matched by some of the 20 templates, and the location of the x's placed by those templates.

The array which contains the x's is used as a mask to determine the extent of the search and the depth is fixed at two moves for each side. The search is actually performed twice, once for white moving first, and once for black moving first. At the end of these searches, it is noted whether either side can force a capture by moving first. This information tells the program whether a move is necessary to cause or avoid a capture.

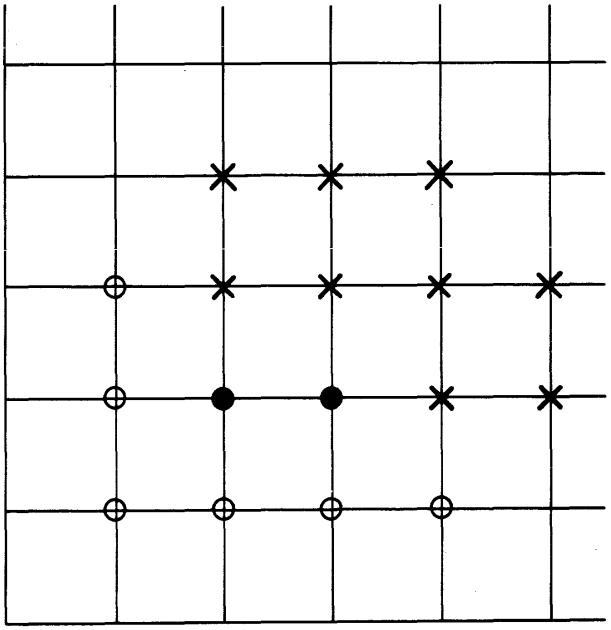


Figure 6—Creation of a mask for lookahead

For example, if black can capture whether he moves first or not, then it is unnecessary for him to move. The decision to move is recorded by placing a weight of 4000 in the array already discussed in connection with the first type of template.

In many cases a depth of two moves is not sufficient to determine whether capture takes place or not. The most common instance of this is known as the "ladder attack" which is illustrated in Figure 7. These situations are characterized by the repeated occurrence of moves which force the opponent to reply or to be captured. In such cases, the search procedure continues to a depth of up to 100 moves to see whether capture finally takes place. No branching takes place during this extension of the look ahead.

When all of the templates have been applied and the heuristic search procedure is through, the program simply chooses the move which corresponds to the highest sum of weights in the array of weights. If the maximum weight is below 100 then the program passes.

This completes the description of the program except for a few minor details of operation. Three seconds are used for the creation of the internal representation and two seconds are used by the template matching procedure. The heuristic search takes from .1 to 4 seconds. A challenger has the option of moving first or second, and can also give the program a handicap of any number of stones on the board. The program is not able to haggle over the final score as GO players often do, hence a referee may be required to supervise the scoring.

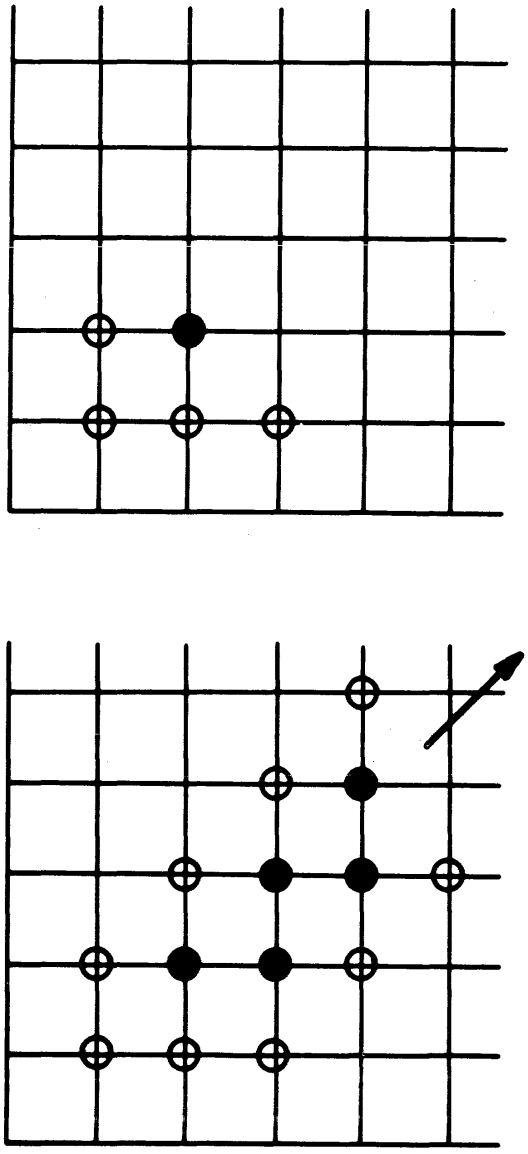


Figure 7—Illustration of the "ladder attack"

RESULTS

The program now has a record of two wins and two losses against human opponents. The opponents can best be described as intelligent adults who know how to play GO, have played from two to twenty games but have not studied the game. The program appears to have reached the bottom rung of the ladder of human GO players. Interested readers are urged to get a GO

board and play through the game listed at the end of this report.

The first type of template, those which imply moves, are responsible for about 80 percent of the moves made by the program. These templates give the program fairly good positional play, especially in the first part of the game.

The remaining templates, together with the look-ahead search, are valuable for avoidance of traps which cause the loss of a stone or two. The opponent's play is also restricted since he must play more carefully. The loss of one stone can have a great effect upon the outcome of the game. The program is able to play without these templates, hence without the search, but opponents soon learn to take advantage of this weakness.

The program plays as if it can "see" such things as the influence of stones, the segmentation of the board, and the armies of black and white stones. This alone makes it a reasonable candidate as a model of visual organization for the game of GO. It would be of interest to test the model by performing standard psychological experiments. For example, a drawing of a GO board could be shown to a human subject with the instructions to segment the board with dashed lines. The results could be compared with the segmentation given by the program. Further work may show how important perceptual mechanisms are to the ability of humans to play GO.

APPENDIX: A GAME BETWEEN THE PROGRAM AND MR G. COWAN

The following game, played between the program and Mr. George Cowan, demonstrates that the program has reached at least the bottom rung of the ladder of human GO players. Mr. Cowan is an undergraduate at the University of Wisconsin, and had five games experience at the time of the contest. The even-numbered moves are by the program (white). Moves which resulted in a capture are indicated by asterisks. The comments are by Professor Paul Purdom, a good GO player, who has shown patient interest in the various GO-playing efforts at Wisconsin.

1.	D 3	Q3
3.	E 9	D17
5.	Q16	H12

too early for such a move

7.	F15	R15

too close to opponent

9.	O16	M 8
11.	R14	H 8

shouldn't give up the corner

13.	M12	C 5
15.	D 5	F12
17.	D 7	J17

still important moves remaining in the corner

19.	M17	R 8
21.	R 5	S 4
23.	D11	K 3
25.	Q11	G 5
27.	Q15	F 6
29.	K16	F17
31.	G16	G17
33.	H16	F 3
35.	E13	E 2

still ignoring upper left side

37.	Q 7	F 8

wasted move

39.	J18	S 7
41.	Q 8	O 5
43.	R 9	P 6
45.	S 8	R 7
47.	R 6	S 9
49.	T 8	T 9
51.	S10	T 7*
53.	T10	R10
55.	Q 9	S11
57.	R11	T11*
59.	S12	O 8
61.	S 6	P 7
63.	T 6	P 9
65.	Q 6	Q10
67.	P10	S10

a free gift from black

69.	O11	Q12
71.	R13	O10
73.	P11	N11

N10 a better move

75.	R12	H17
77.	J16	D 2
79.	P12	B 3
81.	Q13*	B14

at last! C14 better

83.	K17	S17
85.	R16	S15

seems to ignore sacrifices

87.	S14	S16
89.	R17	R18
91.	Q18	F 9
93.	S18	S19
95.	R19*	T17
97.	T19*	T14

both players wasting moves

99.	T13	F13
101.	T16	T15*

103.	T18	G14
105.	T16*	J14
107.	J10	J11
109.	J12	K12
K11 better		
111.	N10	J13*
wasted		
113.	O 9*	P 8
115.	N 9	L14
117.	M14	L13
M13 better		
119.	K15	K 9
121.	L15	M10
123.	M11	L11
125.	N12*	N13
127.	Q 5	M13
129.	M15	O13
131.	N 7	N 6
should connect N8 and cut stone		
133.	O 7	P 5
135.	P14	Q 4
137.	L10	M 9
139.	L 9	K10
141.	L 8	L 7
143.	K 8	J 8
145.	L12	K 7*
147.	K14	E 5
149.	K13	H10
wasted		
151.	N14	P13
153.	F16	E16
155.	E15	D15
157.	C16	C15
159.	E18	E 7
161.	E17	D18
163.	D16*	C17
165.	B17	C18
167.	B18	F18
169.	E19	H18
171.	K18	G19
173.	J19	F19
the program would be much better if it could recognize eye possibilities		
175.	E16	B19
177.	B16	H15
179.	E12	F11
181.	E14	B 7
183.	B 8	C 8
185.	B 9	R 4
187.	C 9	S 5
189.	A 7	T 5*
191.	B 6	D 6
193.	D 4	D 8
195.	C 7*	C 6
197.	B 7	E 8
199.	D 9	C 4
201.	E10	B 5
203.	E11	E 4
205.	F10	G10
207.	G15	G 9
209.	J15	H14
211.	F14	G12
213.	L 4	L 5
215.	L 3	K 4
217.	K 2	M 3
219.	M 2	N 3
221.	M 4	N 4
223.	L 2	N 2
225.	K 5	J 5
227.	M 1	K 6*
229.	J 1	H 2
231.	J 2	J 3

It was agreed to stop the game at this point. The resulting score was: Mr. Cowan . . . 59, program . . . 66, a seven point victory for the program. Approximately 15 minutes of computer time was used, and the entire contest took less than two hours of real time.

ACKNOWLEDGMENTS

This research was conducted with the support of NIH grant MH 12266 and NSF grant GP 7069.

BIBLIOGRAPHY

1 H REMUS
Simulation of a learning machine for playing GO
Proc IFIP Congress 1962

2 E THORPE W WALDEN
A partial analysis of GO
The Computer Journal Vol 7 No 3 1964

3 I GOOD
The mystery of GO
New Scientist January 21 1965 427

4 O KORSCHELT
The theory and practice of GO
Tuttle Rutland Vt 1966

5 E LASKER
GO and GO-MOKO, the oriental board games
Dover New York 1960

6 A SAMUEL
Some studies of machine learning using the game of checkers
IBM Journal of Research and Development Vol 3 No 3 1959

7 A NEWELL
The chess machine
Proc Western J C C 1955

8 C SHANNON
Programming a digital computer for playing chess

Philosophy Magazine March 1950
9 R GREENBLATT D EASTLAKE III S CROCKER
The Greenblatt chess program
Proc F J C C 1967
10 P GREENE
Networks which realize a model for information representation
Transactions of the University of Illinois Symposium on Self-Organization 1961
11 W KOHLER
Gestalt psychology
Liveright New York 1947