Spatial Reasoning in the game of Go

Bruno Bouzy

LAFORIA-IBP Université Paris 6
Tour 46 2ème étage, 4, place Jussieu
75252 PARIS - FRANCE
Tél.: 44 27 70 10 Fax: 44 27 70 00
e-mail : bouzy@laforia.ibp.fr

Abstract

This paper addresses Spatial Reasoning in the game of Go. Combinatorial complexity of the game of Go obliges the Computer Go community to study spatial representations of human players that are complex. These representations contain the concepts of grouping and fractioning. Spatial Reasoning in Go is qualitative. It creates objects and relationships between them in order to qualify them. This description of Spatial Reasoning in Go is a positive contribution to the SR community.

Keywords

Spatial Reasoning, Objects, Relationships, Game of Go.

1. Introduction

Classical reasoning appears to be linked with Natural Language (NL) as human beings are obliged to use words to express their reasonings. NL has been growing between human beings in order to make them communicating, therefore the human body does not provide a communicating tool as efficient as NL is.

But other kinds of reasoning exist : painting, drawing, calligraphy, spatial gesture, dance are good way to express spatial information. Such activities are reasoning activities even if words do not allow to reflect them (correctly or not at all). Spatial Reasoning (SR) opens new perspectives for the study of reasoning, showing that some kinds of reasoning do not depend much on NL.

The purpose of the paper is to show such a kind of reasoning : SR in the game of Go[1]. The paper is structured as follows :

Part 2 shows that beyond the first obstacle of combinatorial complexity, the main obstacle in Computer Go is *spatial* evaluation of positions and, therefore it shows the interest of the paper.

Part 3 shows the SR model of Go. Part 3.1 shows the creation of relevant objects in Go positions and part 3.2 shows the SR about these objects using relationships between them.

Before conclusion, part 4 gives practical results of the full Go playing program INDIGO [Bouzy 1995b] that implements the model [Bouzy 1995].

2. Computer Go needs powerful Spatial Reasoning

[Robson 1982b] has shown that the problem of deciding whether Black (say) can force a win from a given Go position in Go generalized to NxN boards is theoretically of *exponential in time.*
This result is exactly similar to results in other games like Chess [Fraenkel & Lichtenstein 1981] and Checkers [Robson 1982].

[Allis 1994] defines combinatorial complexities of games as follow : E is the number of possible positions in Go and A is the whole game tree complexity. Considering the average length of actual games L and average branching factor B, we get $A = B^L$. Literature on games and Human (H) versus Computer (C) results give the following table :

	Othello	Chess	Go[2]
E	$3^{64} \approx 10^{30}$	10^{43}	$3^{361} \approx 10^{172}$
A	10^{58}	$35^{80} \approx 10^{67}$	$200^{250} \approx 10^{575}$
H v.C	H<C	H≈C	H>>C

These games get increasing complexity and seem to be correlated with Human versus Computer results. "Brute force" succeeds in Othello and Chess but not in Go. Therefore, Go is one of the most exciting challenge for AI [Bradley 1979]. The best Go program is Goliath [Boon 1991]. Our Go playing program is INDIGO [Bouzy 1995b].

---

[1]Go is an oriental game that is very popular in Japan, China and
[2]In average, a 19x19 game lasts 250 moves and there are 200 moves. Of course, this is wrong in the beginning and at the er game but true in average.

This previous combinatorial reasoning is approximative because one could argue that 7x7 Go is as complex as Othello. That is wrong : 7x7 Computer Go results are still very weak on the human scale while best Othello programs are much better than best human players. So, the complexity of Go is not due to the sisze of the board. One suppose that *the complexity of Go is due to the spatial properties of Go positions. Computer Go needs powerful theories about Spatial Reasoning.*

Human Go players are much stronger than the computer. The human skill in Go is mainly due to the intuitive knowledge about space. Therefore, Go programmers must observe human Go players and mimic them. In this meaning, Go computational models are cognitive models. Experiments that have been done on Chess [Chase & Simon 1973] have been done again on Go [Reitman 1976]. The results are similar : experts use "chunks" that allow to overpass novice abilities on actual positions but not on random positions. In our thesis [Bouzy 1995], we elaborated a cognitive model in order to uncover intuitive knowledge about real world. We studied in particular grouping, fractioning, closure, cercling and incrementality.

3. Spatial Representation and Spatial Reasoning in Go

This part shows SR in the full Go playing program INDIGO. This program was the implementation of a cognitive model of Go players. This part uses a mathematical formalism and it is illustrated by examples. Figure 1 shows the start position.

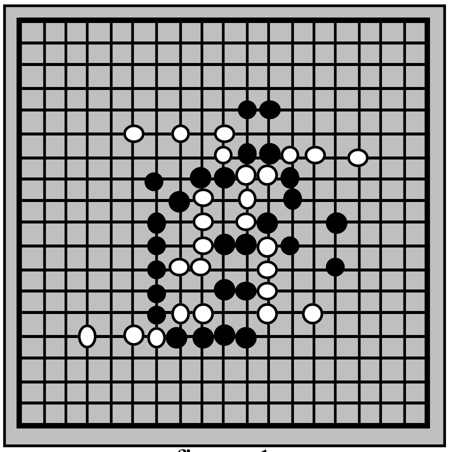

figure 1

3.1 Rules of the game

In this paragraph, we show only the parts of the rules of the game of Go, that are necessary to understand parts 3.2 and 3.3.

Board and intersections

The game is played on a board that contains intersections of N vertical lines with N horizontal lines. Usually N=19. Let *B* be the set of intersections. Each intersection *i* has two coordinates. Let *x(i)* be the horizontal coordinate and *y(i)* be the vertical coordinate.

Neighborhood of an intersection

Each intersection *i* in *B* gets neighbours. Let *up* be the function defined as follows : *j=up(i)* is defined for $y(i)<N$ with $x(j) = x(i), \quad y(j) = y(i) + 1$. One can define three others functions *right*, *down* and *left* on the same template. $V(i) = \{up(i), down(i), left(i), right(i)\}$ is the neighborhood of the intersection i.

Stones

Each player can put a stone on an intersection. Let *color* be the function defining th color of an intersection as follows : $color(i) \in \{empty, black, white\}$. We define the *Empty* set as follows : $Empty = \{i \in B | color(i) = empty\}$. We define the *Black* and *White* sets on the same template.

Connections and strings

The rules of the game define an actual connection between to adjacent intersections of the same color : if $i \in V(j) \quad and \quad color(i) = color(j)$ then *i* and *j* belong to the same *string*. The rules of the game define the black (resp. white) strings of the board B as the connex composants of the sets *Black* (resp. *White*). Let SS_C be the set of the strings of color *c*. Let *SS* be the set of strings. Let $S_c = \{i \in B | \exists s \in SS_c : i \in s\}$ be the set of intersections that belong to a string of color c. We get : $S_{black} = Black \quad and \quad S_{white} = White$.

Liberties

Each string *s* has a set of liberties *liberties(s)* that is defined with : $liberties(s) = dilate(s) \cap Empty$. The *dilate* operator of mathematical morphology [Serra 1982] is defined with : $dilate(x) = \{i \in B | \exists j \in x : i \in V(j)\}$. The *erase* operator is defined with : $erase(x) = \{i \in B | \forall j \notin x : i \notin V(j)\}$. We get : $erase(x) = \neg dilate(\neg x)$ where $\neg x = \{i \in B | i \notin x\}$.

Actions on the board

Each player can put a stone of his color on an empty intersection. The "capture" rule tells that if putting a stone suppresses the last liberty of the string s, the string s is removed from the board.

Aim of the game

The aim of the game is to "control" more intersections than the opponent does. Control of an intersection can be explicit, with a stone on it, or it can be implicit, supposing that the opponent cannot put a stone on it without being captured. *Game of Go is a race to fill the space. Therefore SR is crucial in Go.*

3.2 Spatial representations of the human player

This paragraph adresses SR at the human player level. SR of the human Go player has two levels : an "elementary" level where elementary objects are recognized with direct pattern-matching and an "iterative" level where complex objects are recognized through a complex SR.

Elementary level

This level is a collection of patterns that are useful to build objects of the iterative level.

Pattern

A pattern is a "small" (<25) connex set of intersections. A pattern gets friend intersections, opponent intersections and empty intersections. A pattern has a color which is the color of the friend intersections. A pattern has a state. According to the spatial locations of the friend, opponent and empty intersections the state of the pattern can be : Positive, Negative or Fuzzy. A pattern has one or several types : *connection*, *divider* or *contact*. The computational model contains a pattern database. At run-time, a pattern-matching algorithm detects actual patterns that are present on the board.

Connections and dividers

Let $CC_C(S)$ be the set of connections of color c whose state is S. Let $DD_C(S)$ be the set of dividers of color c whose state is S. Figure 2 shows six patterns that are both connections and dividers in different states.

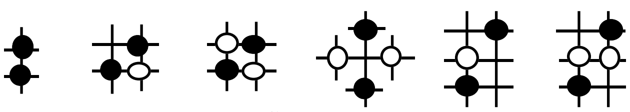

figure 2

Contacts

Let CT be the set of contacts. Figure 3 shows patterns that are elementary contacts.

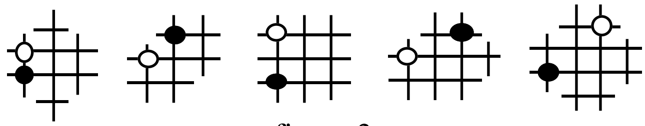

figure 3

These patterns are used in the next level to build iterative objects.

Iterative level

The iterative level is a collection of objects that are built iteratively using patterns of the elementary level. Iterative objects are "groups" and "fractions".

Iterative objects

An iterative object is a connex set of intersections that is constructed iteratively. One iteration of the construction depends on the type of the objects. Iterations stop when the object is the same after one more iteration. An iterative object has a color.

Groups

At the beginning, the groups of a given color are initialized with the strings of this color. Then iterations are performed until they do not change the set of groups.

One iteration

Let *g* be the group in construction. Let *CC(g)* be the set of connections of the same color as g whose state is Positive. We get : $CC(g) = \left\{ cc \in CC_{color(g)}(Positive) \middle| cc \cap g \neq \emptyset \right\}$.

We define *N(g)* as the positive friend neighborhood of g as follows : $N(g) = \left\{ h \in GG_{color(g)} \middle| h \neq g \quad and \quad h \cap CC(g) \neq \emptyset \right\}$. After one iteration, g is replaced with g and the groups that belong to *N(g)*. We write : $g \to g \bigcup_{h \in N(g)} h$

Notations

Let *GGc* be the set of groups of color c staying on the board. Let G_C be the set of intersections where there is a group of color c. We get : $G_c = \left\{ i \in B \middle| \exists g \in GG_c : i \in g \right\}$.

Figure 4 shows the groups that correspond to figure 1.

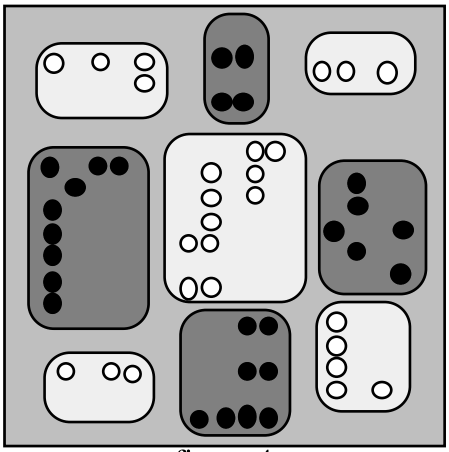
figure 4

Fractions

Let F_C be the set of the intersections that are not in a divider. We get : $F_c = \{i \in B | \forall d \in DD_c(Positive): i \notin d\}$. Let FF_C be the set of the connex composants of F_C. Let us call an element f of FF_C a *fraction* of color c.

Conclusion

We defined groups, and fractions in a constructive way. To build these objects we used
operators of the set theory : $\cup$ and $\cap$,
topological properties of connection and division.

3.3 Spatial reasoning or how to qualify groups

Aim of SR in Go

The aim of spatial reasoning in Go is to give an evaluation *E(B)* of the board B. It sums the contributions *C(x)* of each group of the board :

$$E(B) = \sum_{x \in GG} C(x) sign(x)$$

Each contribution *C(x)* of a group x is defined as follows : $C(x) = \begin{cases} 0 & if \quad O(x) = Fuzzy \\ size(x) & if \quad O(x) = Positive \end{cases}$

and $sign(x) = \begin{cases} +1 & if \quad color(x) = Black \\ -1 & if \quad color(x) = White \end{cases}$

O(x) is defined below.

SR is a loop of iterations.

One iteration

During one iteration *opponentship O(x)* of all groups x are computed. O(x) gathers information about the relationships between the group x and its opponent neighbours. To compute *O(x)* of a group x, our system computes *cercling* - or *emptyship* - *E(x)* before. E(x) gathers information about the relationship between the group x and the fraction in which x stays.

Emptyship or Cercling of a group

Each group of a given color is included in a fraction of the opposite color. Conversely, a fraction of a given color contains zero, one or several groups of the opposite color. Let us define the *cercling distance* of a group as the smallest number N such as : $\Delta^{N+1}(g) = \Delta^N(g)$ with $\Delta(g) = dilate(g) \cap F_{\neg color(g)}$. We can fix a threshold T such as N<T means that g is *cercled* and that g is not cercled in the converse case.

Let *z = fraction(x)* be the fraction containing x and N the cercling distance of x by z. We define Cercling or Emptyship *E(x)* as follows :

$$E(x) = \begin{cases} Negative & if \quad N < 2 \\ Fuzzy & if \quad N = 2 \\ Positive & if \quad N > 2 \end{cases}$$

Figure 5 shows Emptyship of each group on the board. E means emptyship. > means Positive and < means Negative.

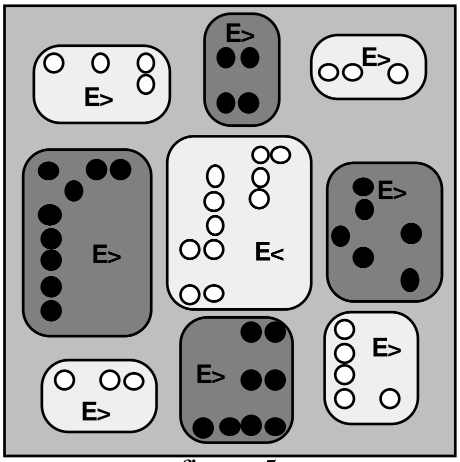

figure 5

Opponentship of a group

Let us define the opponent neighborhood of a group x as the set that is defined as follows :

$$\text{opponent} - \text{neighborhood}(x) = \{y \in GG_{\neg color(x)} | \exists c \in CT: c \cap x \neq \emptyset \quad \text{and} \quad c \cap y \neq \emptyset\}$$

Given two opponent groups x and y, we define a relationship *R(x,y)* that tells which group from x and y is the stronger one. *R(x,y)* depends of values *E(x)*, *E(y)*.

To compute R(x,y), we use the following rules :

$$(\exists\alpha \in \{x,y\}: E(\alpha) = \text{Fuzzy}) \Rightarrow (R(x,y) = \text{Fuzzy})$$
$$(E(x) = \text{Negative} < E(y) = \text{Positive})$$
$$\Rightarrow ((R(x,y) = \text{Negative}) \wedge (R(y,x) = \text{Positive}))$$
$$(E(x) = E(y) = \text{Positive}) \Rightarrow (R(x,y) = \text{Fuzzy})$$
$$(E(x) = E(y) = \text{Negative}) \Rightarrow (R(x,y) = \text{Fuzzy})$$

Figure 6 shows the relationships between adjacent groups.

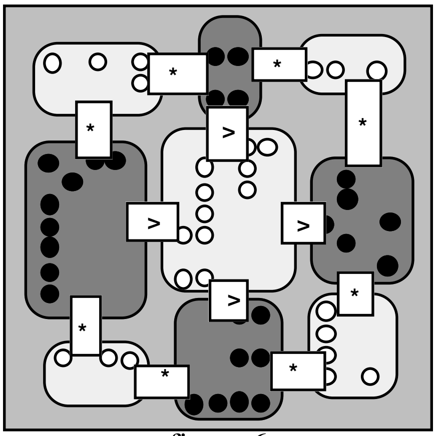

figure 6

If all relationships *R(x,y)* are Negative, then opponentship is also Negative :

$$(\forall y \in \text{opponent} - \text{neighborhood}(x): R(x,y) = \text{Negative})$$
$$\Rightarrow O(x) = \text{Negative}$$

If one interaction with its opponents is Fuzzy, then opponentship is also Fuzzy :

$$(\exists y \in \text{opponent} - \text{neighborhood}(x): R(x,y) = \text{Fuzzy})$$
$$\Rightarrow O(x) = \text{Fuzzy}$$

Figure 7 shows opponentship of each group on the board. > means Positive, * means Fuzzy and < means Negative.

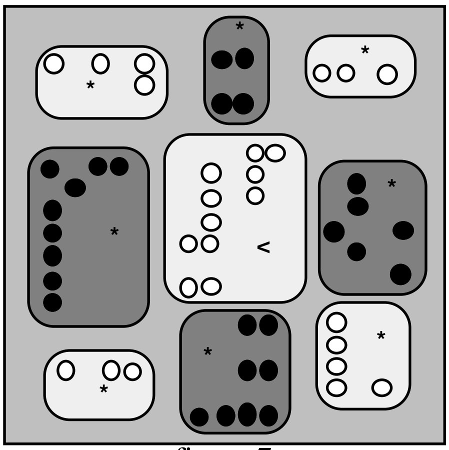

figure 7

The fusion operator

Classical closure operator of mathematical morphology [Serra 1982] applies on objects with only one color. We use a generalization of this operator that manages objects with two colors and that smoothes the groups at a fixed scale. This generalization is described in [Bouzy 1995]. Let us call it *closure*. Then we define the $\coprod_*$ operator as a generalization of the $\coprod$ operator. Let A and B be two closed sets, we get : $A \coprod_* B = closure\left(A \coprod B\right)$.

Death of a group

When a group x gets *O(x)=Negative*, Go players say "the group is dead". Conceptually, a new group Y is created by fusion of the opponent groups around the group x :

$$Y = x \cup_* \coprod_{y \in opponent-neighborhood(x)}^{*} y$$

When *Y* is created, the set *G* of all the groups on the board also changed. A new iteration starts again on G. Figure 8 shows the new set of groups.

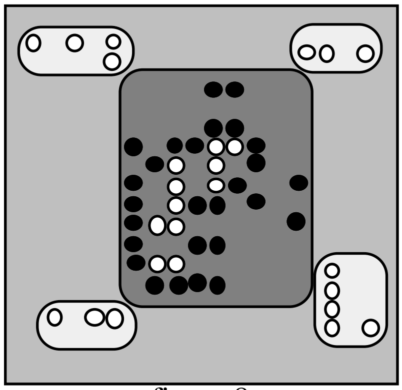

figure 8

The group x is *in* the group Y. Objects that are built by SR can be composed of sub-objects.

End

SR ends when no group with Negative opponentship is detected. Then, *E(B)* is computed.

4. Implementation, results and discussion.

4.1. Implementation - Results in practice.

The Go playing software INDIGO is an implementation of this model with C++ code and specific diagrammatic language (500 rules). INDIGO is ranked 4th upon 7 on the 19x19 Computer Go Ladder [Pettersen 1994] and 8th upon 11 on the 9x9 ladder. It has been developped in two years between fall of 1991 and spring 1994 in connection with the cognitive model that we built.

4.2. Discussion

In this paragraph, we link SR in Go with other fields of the SR community in order to initiate an interdisciplinary dialogue.

Mereology

Our work is an illustration of mereology [Lesniewski 1927-31]. First, our system deals much with *part-of* relationships. An intersection is part of a string, that is part of a group. The whole board is composed with fractions. Second, we defined two fusion operators : $\cup$ and $\cup^*$ that are very useful to build objects.

Using terminology about contacts described in [Aurnague & Vieu 1995], we see that the new object z that is created with the *fusion* operator using the dead group x and the opponents groups y, is composed of sub-objects that are parts of it. The contacts between the dead group x and the opponent groups y are *weak* before the fusion but become a *strong contact* after it. In the general case, the contacts between opponent groups whose states are not Negative can be seen as a *weak contacts.*

Qualitative reasoning and quantitative reasoning in Go

The part of SR that is presented in this paper is qualitative. But another part of SR in Go is quantitative. At the move decision level, succeeding to reach a goal on an object of size N gives a value of N times the value of succeeding to reach a goal on an object of size 1.

Propositional versus diagrammatic reasoning

In our model, SR uses diagrammatic formalism (see patterns of the elementary level) or propositional formalism (see the iterative level).

Connection between SR in Go and NL

Our system builds a semantic net. With this information, it would be possible to make our system explaining in NL its way of playing. It would be an illustration of the new topic of integration of NL and vision. [Herskovits 1986] has hown that spatial prepositions are crucial in spatial discoures understanding even if it is difficult to model them. The main prepositions that are used in comments in Go are : 'up', 'down', 'left', 'right', 'middle', 'edge', 'corner', 'in', 'touch', 'cercled'. This will be the subject of another paper.

GIS

The CBM [Clementini & De Felice 1995] uses five operators. Let us see which one are used in our work.

touch operator : yes (*contacts* of mereology).

in operator : yes (after a *fusion* operation, a group is *in* another one).

cross operator : yes (a group is crossing the fraction of opposite color that contains it).

overlap operator : yes (two fractions of opposite color are *overlapping* each other).

disjoint operator : yes (two different groups).

boundary for an area : yes (the *dilation - identity* operator).

boundary for a line : no (distinction between line and area is not useful in Go).

So most of operators of CBM are sound for our Go model.

In our model, the following operators are also useful : the *closure* operator and the *cercling* operator. Perhaps, they could be also useful in GIS.

CM

Our model is structured in levels as general CM are. In this paper, we focused only on the "iterative object" level. Other levels are described in [Bouzy 1995]. Our "iterative object" level corresponds to "conceptual spatial object level" in [McMaster & Barnet 1993] and to "geometrical level" in [Aurnague & Vieu 1995].

5. Conclusion

In this paper, we show SR in the game of Go. Complexity of this game is high enough to forbid combinatorial approach and actual results between the man and the computer oblige to study human SR.

Our work about Go was presented in a mathematical formalism. We defined objects that are familiar to our intuition : groups and fractions. We used operators such as fusion, overlap, touch, in, proximity, cercling and closure. We have shown how SR in Go allows to evaluate the whole board. Our program uses this model when it plays full games of Go. It is ranked on the international computer Go ladder. Therefore a computational model of SR is very useful in a Go playing program.

In showing a kind of SR where words are not so important, we hope that this description will be a positive contribution to the Spatial Reasoning community.

6. Bibliography

[Allis 1994] - Allis L. V. - Searching for Solutions in Games and Artificial Intelligence - PhThesis - Vrije Universitat Amsterdam - Maastricht.

[Aurnague & Vieu 1995] - Aurnague M. and Vieu L. - Semantics an Pragmatics of Natural Language : Logical and Computational Aspects. - ILCLI Series, 1, pp. 69-126 - Donostia (San Sebastian), Spain - 1995.

[Boon 1991] - M. Boon - Overzicht van de ontwikkeling van een Go spelend programma - Afstudeer scriptie informatica onder begeleiding van prof. J. Bergstra.

[Bouzy 1995] - B. Bouzy - Modélisation du joueur de Go - Thèse de doctorat d'informatique de l'université Paris 6

[Bouzy. 1995b] - Bouzy B. - The INDIGO program - Proceedings of the 2nd Game Programming Workshop in Japan - pp.191-200.- Kanagawa1995.

[Bradley 1979] - M.B. Bradley - The game of Go, the ultimate programming challenge? - Creative computing, 5, 3, pp. 89-99 - Mars 1979.

[Chase & Simon 1973] - Chase W. Simon H. - Perception in Chess - Cognitive Psychology. - 4, pp. 55-81 - 1973.

[Clementini & De Felice 1995] - Clementini De Felice - A Comparison of Methods for Representing Topological Relationships - Information Sciences - 3, pp. 149-178 - 1995.

[Fraenkel & Lichtenstein 1981] - Fraenkel A.S., Lichtenstein S. - Computing a perfect strategy for n by n Chess requires time exponential - Journal of Combinatorial Theory, Serie A, Vol. 31, N°2, September 1981, pp. 199-214

[Herskovits 1986] - Herskovits A. - Language and Spatial Cognition. An interdisciplinary Study of the prepositions in English. - Cambridge University Press, Cambridge, MA.

[Kohler 1969] - The task of Gestalt Psychology - Princeton University Press - 1969

[Kuipers 1978] - Kuipers B.J. - Modelling spatial knowledge - Cognitive Science, 2, pp. 129-153.

[Lesniewski 1927-31] - Lesniewski S. - O podstawach matematyki - Przeglad Filosoficzny, vols. 30-34

[McMaster & Barnet 1993] - McMaster R. Barnet L.- A spatial-object level organization of transformations for cartographic generalization - Proceedings 11th Auto-Carto Conference - Minneapolis - MN, Oct. 1993, pp. 386-395.

[Pettersen 1994] - The Computer Go Ladder - http://cgl.ucsf.edu/go/ladder.html

[Reitman 1976] - Reitman J.S. - Skilled perception in Go: deducing memory structures from inter-response times - Cognitive Psychology, 8, 3, 1976.

[Robson 1982] - Robson J.M. - N by N Checkers is exptime complete, TR-CS-82-12, Australian National Unviversity Department of Computer Science - 1982

[Robson 1982b] - Robson J.M. - The Complexity of Go, TR-CS-82-14, Australian National Unviversity Department of Computer Science - 1982

[Serra 1982] - J. Serra - Image Analysis and Mathematical Morphology - Academic Press - London - 1982.