Strategic Search:A New Paradigm for Complex Game Playing. Application to the Game of Go

Régis Moneret
LIP6
Université Pierre et Marie Curie
4, Place Jussieu, 75005 Paris, France
E-mail: Regis.Moneret@.lip6.fr

Abstract

In this article, we describe a new search technique that we call "Strategic Search": Instead of developing a game tree by considering every legal moves in a given position, we consider only the goals that can be achieved from that position. As achieving a goal can take tens to hundreds moves in complex games like Go, this means that the search will run much deeper, than with usual search techniques. Of course, to do this, one has to be able to predict and assess the future position once the goal will be achieved. This is done by applying modeling rules that are at the time being, hand-coded in the system, but should be learned in the future by a machine learning module.

1. Introduction

To win a war or a match in a game of Go, one has to win battles and one has to choose which battles to fight. Winning battles is the concern of tactics. Choosing the battles to fight, the concern of strategy. Being a skilled tactician or a skilled strategist alone is not enough. Only a balance of both can lead to victory. Although this conclusion has been clear since the ancient Greeks coined both terms, strategy and tactics have had opposite fates up to now.

Because tactical plans are easy to spot, because the way the resources are employed in order to achieve tactical aims can easily be explained, many programs ([Pitrat, 1971] , [Minton, 1984] , [Cazenave, 1996] for instance) have already been devised to find tactics in games. On the contrary, because strategy is a choice between tactics, because it is stealthy, and because its effects can only be perceived long after the original decision has been made, few systems have dealt with the very issue of strategy yet. This article presents a way to remedy this lack, and to integrate strategy and tactics in the same system: Strategic Search.

In Section 3, we will present the basic principles that underlie Strategic Search. To do so, we will provide an example from the Asian Game of Go which will be briefl described in Section 2. In Section 4, we will then try to define the limitations of Strategic Search and the cases where this approach can be useful. We will then conclude this article by presenting results of the system we have designed, and by relating our work to previous systems dealing with the issue of strategy.

2. The Game of Go

Go evolved out of a method of divination used in China 4000 years ago. It is a two-player, information complete, deterministic game between Black and White on a 19x19 square grid, the Go board. The board is entirely empty at the start. Each player places stones of his color, one per turn, on any empty intersection, trying to enclose regions of empt intersections touched only by his stones. Such regions are called "territories". Once played, stones do not move from their original spot, except when they get entirely surrounded by enemy's stones. In such cases, they are taken prisoner and are removed from the board. Whoever controls more territories and prisoners when the game ends, wins.

Go Programming dates from 1961. Since then, programs have made great strides, but they are still at the level of advanced beginners. This backwardness compared with Chess programs, for instance, is mainly due to the extreme complexity of Go (the average size of a game-tree is about 1E+250 nodes for Go compared with 1E+50 only for Chess). This complexity prevents the successful use of "brute-force" search techniques to solve this game, which makes it a thought-provoking challenge for Artificial Intelligence.

3. Strategic Search

Strategic Search is a new kind of heuristic search (Pearl, 1981]), whose principle is to replace the search in the game tree by a search in an abstract tree (that we call the Strategic Tree), which is much smaller, since a single branch in the Strategic Tree may represent several moves in the original game tree. Once the Strategic Tree of the current position has been developed, goals are classified according to thei interest in the given position, and the best one is considered to be the "strategic theme" to be played.

Linked to each goal, but not represented on Figure 3, are a certain number of tactical plans (= classical plans with preconditions) as shown in Figure 4. The plans linked to best goal are then simulated to check whether the strategic theme of the position can be actually achieved and, if so, with what moves. If every tactical plan associated to the current best goal has failed, that goal is then considered to be unattainable, and the second best goal is selected to be the "strategic theme", its tactical plans checked and so on, until one goal is proved to be achievable.

To explain this better, let us consider the example of Figure 1:

3.1. Instantiation of the Goal Tree

As we have already said, Figure 3 shows a part of the tree that has been generated from the rules of Go given to the system. This has been done quite simply by following the dependencies between the concepts (e.g. 'territories', 'eyes', 'strings') expressed in the rules of the game.
The goal tree can be seen as a hierarchy of class, that will be instantiated in the position of Figure 1. This is namely the first step of Strategic Search: The system starts from th root node of the goal tree and checks whether the current node is associated to modeling rules (see Figure 4). If not, it checks its children recursively. If the current node possesses modeling rules, the system tries then to match these rules on the current position. If it fails, it goes to the node's childre as before. If some of the rules do match, they provide information about what the position should look like if the current goal is achieved.

3.2. Modeling Rules

Suppose that in this example, the system has only fou modeling rules :

- R_CONNECT associated to the goal CONNECT STRINGS, which tells that when two friendly stones delimit a segment of length between 5 and 10 (= a *sector line*) intersection, which is not cut by enemy stones, the goal can be achieved, and the resulting position consists of friendly stones regularly placed on that segment.

- R_DISCONNECT associated to the goal DECONNECT OPPONENT STRINGS. It the inverse of R_CONNECT. Instead of placing friendly stones on a friendly sector line, one places friendly stones on an enemy sector line.

- R_KILL associated to the goal KILL ENEM STRINGS, which tells that enemy strings with less than 5 liberties can be captured, and that the resulting position consists of the enemy stones being cleared from the board and added to the prisoners.

- R_PROTECT associated to the goal MAKE STRINGS. It is the inverse rule of R_KILL, which tells that friendly strings with less than 5 liberties should be protected, and that the resulting positio consists two eyes being added to the string.

3.3. The Construction of the Strategic Tree

Once a modeling rule has matched, the new position is then placed in a tree structure, the Strategic Tree (see Figure 6). A search technique (AlphaBeta in the present case), can then be applied to determine the best sequence of goals to be played (in Figure 1, CONNECT STRINGS (Y)).

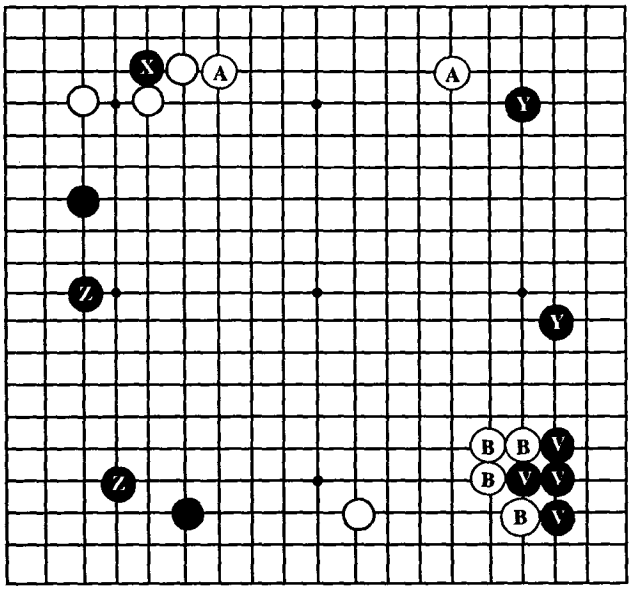

Figure 1: Example of Go game (Black to Play)

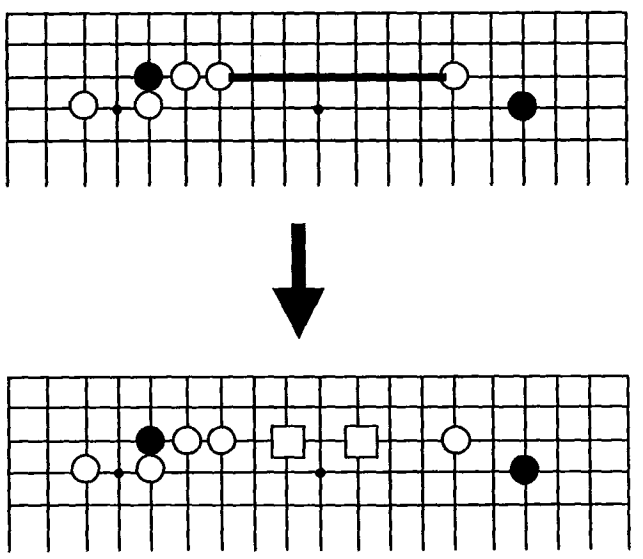

Figure 2: An Instantiation of R_CONNECT

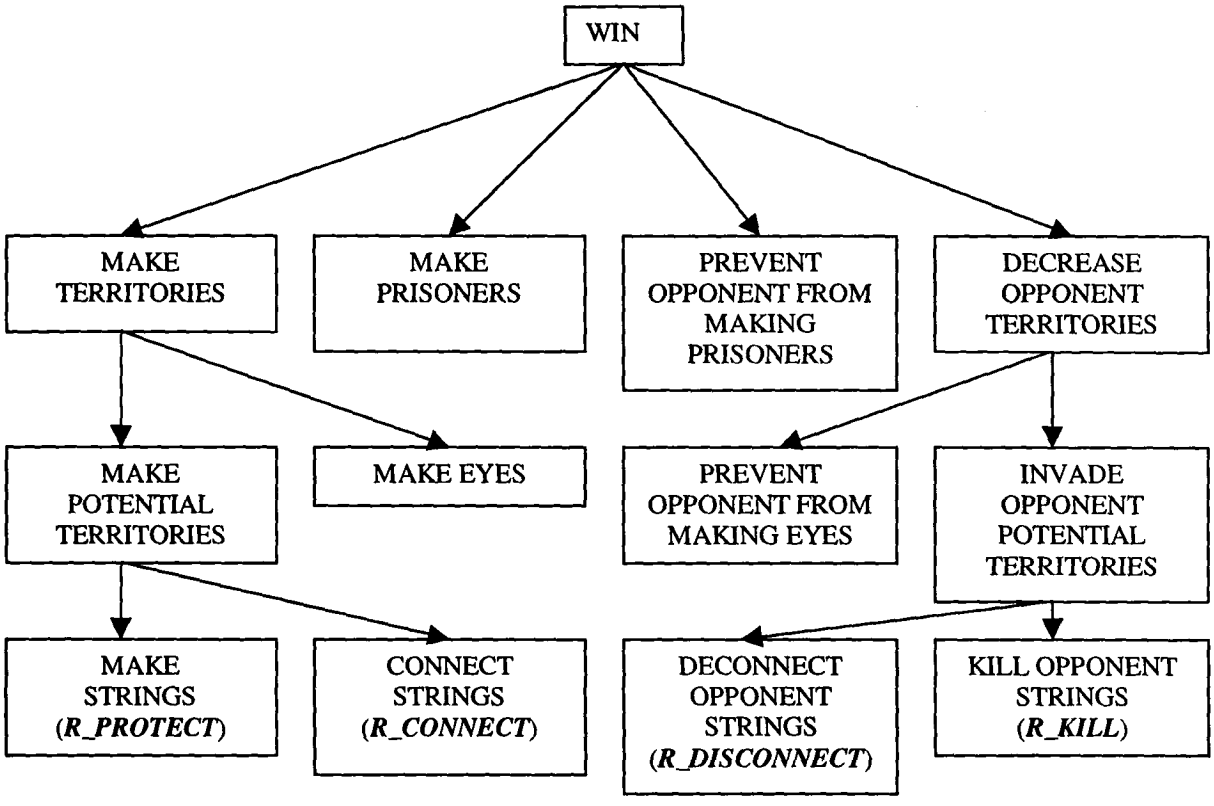

Figure 3: Part of the Goal Tree generated automatically from the rules of Go (the modeling rules are in bold italic, and the names of the different goals have been rewritten for better readability)

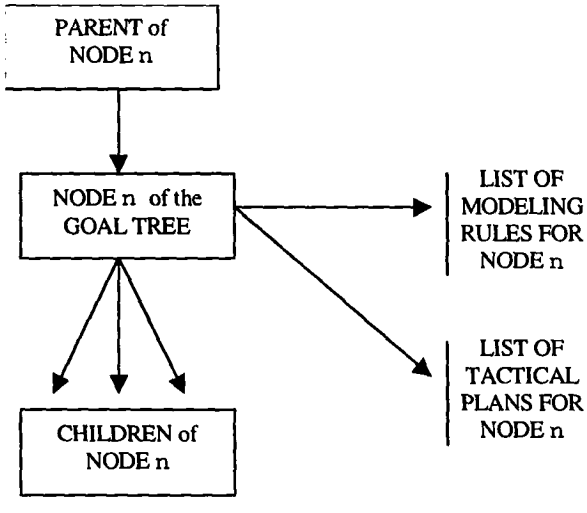

Figure 4: Zoom of a Node of the Goal Tree

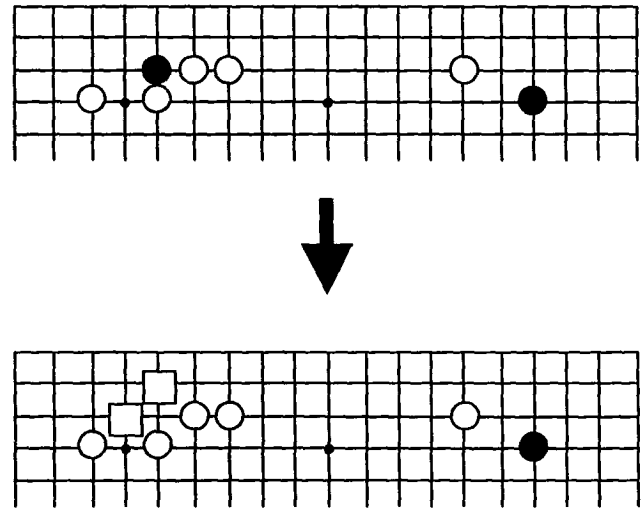

Figure 5: An Instantiation of R_KILL

4. Evaluation of Strategic Search

4.1. Estimation of the Errors that May Occur During the Search

As can be seen in the previous example, the quality of the search is mainly determined by the quality of the modeling rules. Four different causes of error can be distinguished

- An important goal is missing in the goal tree. This is not a serious mistake in games like Go where human experts can easily spot and correct such mistakes.

- An error is caused by the evaluation function used to assess the value of the terminal positions. As this error is common on most search techniques, this is not a serious limitation of Strategic Search.

- A modeling rule is missing : a goal can be achieved from the given position, but the system does not realize it. The system finds a reasonable goal to achieve, but not the optimal one.

- An error is caused by a spurious modeling rule.

From all these kind of error, only the last one is really detrimental to the quality of the search. It cannot be avoided, but its consequences can be limited : since th modeling rules may be spurious, if we apply another modeling rule on a part of the Goban that has already been transformed, we take the risk of multiplying the errors. To prevent this, **we can impose that the modeling rules should be matched solely on parts of the board which have not been altered yet.** In practice, this would account to limiting the depth of the search to 2 or 3 plys. Of course, this can be done more easily on opening games than on middle or end games, when every part of the board interact with each other.

4.2. Estimation of the Benefits of Strategic Search

In the case of Figure 1, considering that the average depth for achieving each goal is at least equal to three moves, we have replaced a game tree of about $361^{2x3} \approx 2{,}21E15$ nodes by a strategic tree of about 6x5 = 30 nodes (after having achieved goal G, the goal -G is no more achievable). More generally, if 'n' is the average number of achievable goals in a given position, 'depth', the depth of the search, and 'p' the average depth for achieving a goal, the game tree is about $361^{pxdepth}$ nodes whereas the strategic tree has ^{depth} nodes where n is generally << 361. Let us suppose that n = 100, which is quite considerable, and p = 4, which is little. The ratio 'r' is the equal to $361^{xdepth} / 100^{depth}$. If we take the logarithm, Log(r) = depth x (4xlog(361) - log(100)) ≈ 8,2 x depth. Hence $r \approx 10^{8,2}$. For a low depth, like 3, $r \approx 1\ ^{24}$.

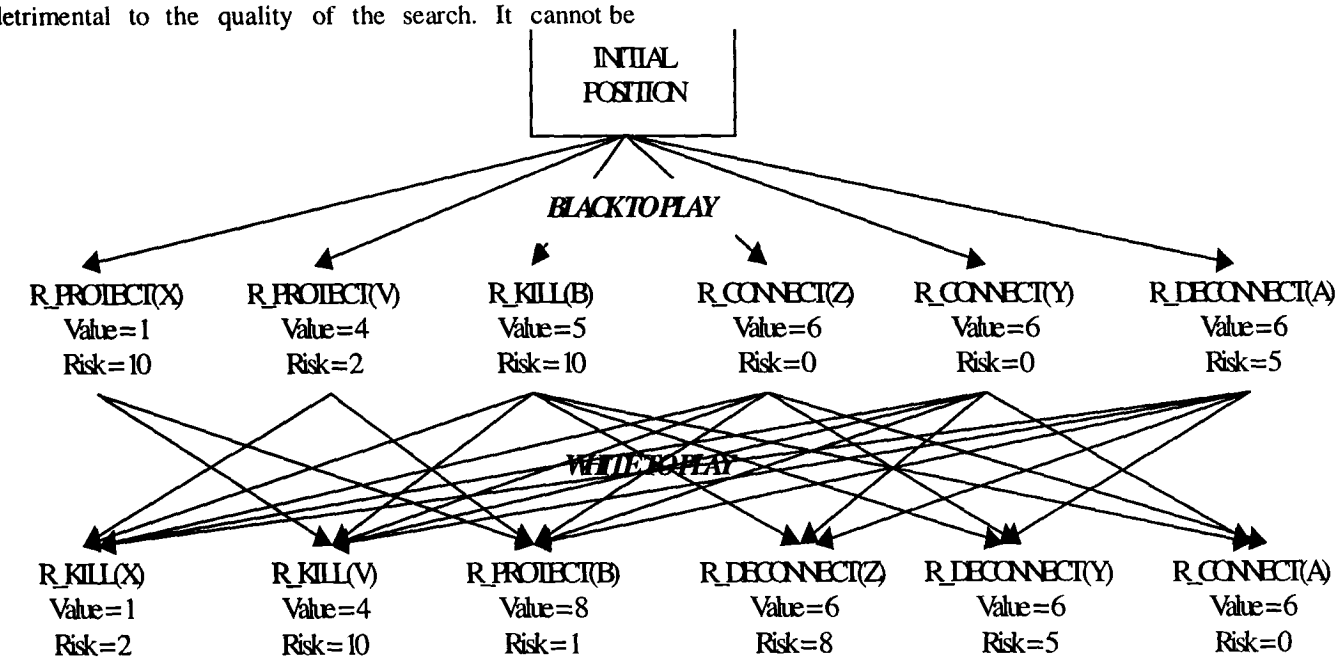

Figure 6: Part of the Strategic Tree derived from the position of Figure 1 and from the Goal Tree of Figure 3 (the arguments of the rules refer to the names of the stones in Figure 1)

4.3. Other Benefits of Strategic Search

Unlike other search techniques, Strategic Search is perfectly suited to human comprehension. Moreover, even if the results may be under optimal, the system does not seem to wander aimlessly. On the contrary, its moves are always intelligible, and based on global considerations about the current position.

4.4. Strategic Search and Incomplete Information

Deterministic games with complete information may seem poor models for studying search under incomplete information. However, one should not forget **the price** of this information: brute force searching of the game tree in Go for instance, would take more than a lifetime, and th result might be not even found then !! So, to be effective, any search technique in games like Go must sacrifice the completeness of the information for the sake of speed. The basic idea behind Strategic Search, is to pay a (relatively) small price to find the strategic theme of the position before paying the full price to check locally whether this theme can be achieved. Moreover, to prevent both the exponential growth of the working memory (as in best-first search for instance), and the cumulating of errors, as was said in 4.1, we limit the depth of search in the Strategic Trees to a fe plys (which amounts to a few tens of plys in the original game tree)

5. Conclusion

In this article, we have introduced a new search paradigm, called Strategic Search, which replaces the search in the game tree, with a search in the space of the achievable goals from a given position. Like many heuristic search, Strategic Search requires a great amount of knowledge, which has been hand-coded at the time being. We have namely written about 40 modeling rules for the goals shown in Figure 3. This has enabled us to check our approach on Fuseki tests taken from [Miyamoto, 1991]. Although it is difficult to judge the results, since our system, lacking tactical knowledge, only suggests the first best strategic theme (and not the best move to satisfy this theme), we were pleased to see that in 12 tests on 20, it found the best goal to pursue. In the other cases, modeling rules were found either to be lacking or to be over-confident (about the ability to kill or save groups for instance). It seems to us that Strategic Search as it is now is well adapted to periods of the game like Fuseki (= opening) when modeling the result of goals is easier.

Our next step is now, to devise a learning module that will generate the modeling rules and the tactical plans automatically from human games, and test the system on others parts of the game (namely the middle game)

6. Related Works

Search, as related to problem-solving, has always been a corner-stone of Artificial Intelligence Research. Since [Shannon, 1950] first thought of using a heuristic function to approximate the true value of a position and developed min-max search, a brute-force search technique which was later improved by $\alpha\beta$ pruning ([Edwards and Hart, 1963], [Knuth and Moore, 1975]), much progress has been made in the field. At the beginning, best-first search ([Stockman, 1979] , [Berliner, 1979], [McAllester, 1988], [Allis, 1994]) were developed to try to limit the exponential growth of the searched nodes, but they had a considerable drawback, since the require a working-memory exponential in the depth of the tree. Then, as pathological games were discovered ([Nau, 1982]), researchers went on to consider alternatives to minimax that would abandon some of its assumptions. Selective Search techniques using Limited-Rationality and Probabilities ([Pearl, 1988], [Baum, 1993]) were developed in order to use the decision-theoretic concept of expected utility to define the value of game-states and the usefulness of reasoning actions. But the result was that, instead of spending time searching the tree, these methods spend it at the meta-level, to decide where to search. Strategic Search bypass this issue, since the decision procedure, being coded in the modeling rules can be learned "off line". At run time, the matching of the decision rules, although it does take some time, is generally quite fast, since the rules are hierarchically organized, and since, on average, only fraction of the entire set of rules are actually tested.

Another technique, Abstract Search ([Frank, 1994]) was introduced. Although abstraction in planning and game-playing was not, by far, a new idea, this technique proved particularly interesting for strategy planning ([Ricaud, 1997]). But, as the abstract states are generally a class of "concrete" states sharing a certain set of features, the choice of these feature has a considerable impact on the search quality. Moreover, once a path to the solution has been found in the abstract space, it must be checked to see whether it is actually correct in the "concrete" situation. Strategic Search can also be seen as a kind of abstract search, but as the result of applying modeling rules to a position is also a "concrete" position and not an abstract position, we do not have the problem of choosing the features to represent the abstract states. Moreover, it is quite easy to improve the quality of the search since we only need to provide the system with more modeling rules.

7. References

[Allis, 1994] Louis Victor ALLIS *Searching for Solutions in Games and Artificial Intelligence* Ph.D. Thesis, Department of Computer Science, University of Limburg, Maastricht, 1994.

[Baum, 1993] Eric B. BAUM On Optimal Game Tree Propagation for Imperfect Players. AAAI, pp.507-512, 1993.

[Berliner, 1979] H. J. BERLINER *The B^* Tree Search Algorithm: A Best-First Proof Procedure.* Artificial Intelligence (12), pp.23-40, 1979.

[Cazenave, 1996]Tristan CAZENAVE *Explanation-Based Learning in Games : an Application to the Game of Go.*

Proceedings of the 8th International Conference on Artificial Intelligence Applications (EXPERSYS-96), Paris, 1996.

[Edwards and Hart, 1963] Daniel J. EDWARDS and Timothy P. HART *The $\alpha\beta$ Heuristic.* Artificial Intelligence Project Memo 30, MIT, Massachusetts, 1963.

[Frank, 1994] Mickael P. FRANK *Advances in Decision-Theoretic Artificial Intelligence: Limited Rationality and Abstract Search.* Master of Science, MIT, 1994.

[Knuth and Moore, 1975] D.E. KNUTH and R.W. MOOR *An Analysis of Alpha-Beta Pruning.* Artificial Intelligence (6), pp.293-326, 1975.

[McAllester, 1988] D.A. MCALLESTER *Conspiracy Numbers for Min-Max Search.* Artificial Intelligence (9), pp.37-48, 1988.

[Minton, 1984] Steven MINTON. *Constrained-Based Generalization: Learning Game-Playing Plans from Single Examples.* Proceedings of the Fourth National Conference on Artificial Intelligence, pp.251-254, 1984.

[Miyamoto, 1991] N. MIYAMOTO *Test Your Go Strength .* Ishii Press, 1991.

[Nau, 1982] Dana S. NAU *An Investigation of the Causes of Pathology in Games.* Artificial Intelligence 19(3), pp.257-278, 1982.

[Pearl, 1981] Judea PEARL *Heuristic Search Theory: A Survey of Recent Results.* IJCAI, pp. 554-562, 1981.

[Pearl, 1988] Judea PEARL *Probabilistic Reasoning in Intelligent Systems: Network of Plausible Inference.* Morgan Kaufmann, San Mateo, 1988.

[Pitrat, 1971] Jacques PITRAT. *A General Game-Playing Program.* In Artificial Intelligence and Heuristic Programming. Findler and Meltzer eds., Edinburgh University Press, pp. 125-155, 1971.

[Ricaud, 1997] Patrick RICAUD *A Model of Strategy for the Game of Go Using Abstraction Mechanisms.* International Joint Conference on Artificial Intelligence, vol.1, pp.678-683, 1997.

[Shannon, 1950] Claude E. SHANNON *Programming a Computer for Playing Chess.* Philosophical Magazine, XLI(314), pp.256-275, 1950.

[Stockman, 1979] G. STOCKMAN *A Minimax Algorithm Better than Alpha-Beta ?* Artificial Intelligence (12), pp. 179-196, 1979.