

Canonical Sequence Directed Tactics Analyzer for Computer Go Games

Chung-Chih Li
Computer Science Department
Lamar University
Beaumont, Texas 77710, USA

Hikyoo Koh
Computer Science Department
Lamar University
Beaumont, Texas 77710, USA

Abstract

We present an approach used in CSDTA (Canonical Sequence Directed Tactics Analyzer) that uses canonical sequences (Joseki) in hoping to improve computer Go programs. We collect 1278 canonical sequences and their deviations in our system. Instead of trivially matching the current game to the collected sequences, we define a notion of similarity to extract the most suitable move from the candidate sequences for the next move. The simplicity of our method and its positive outcome make our approach a promising tool to be integrated into a complete computer Go program for a foreseeable improvement.

1. Introduction

Go is a two-player board game using identical pieces called stones (each player plays a color, black or white) to occupy territory on the board.¹ The game has been very popular in some Asian countries for centuries. In the past few decades, Go has gradually caught attention in the West. As a matter of fact, to program a computer for playing Go at an acceptable strength has been one of the most challenging tasks in Artificial Intelligence [4]. After Shannon's first call in [13], computer scientists took "only" fewer than 50 years to have IBM Deep-Blue [6] that defeated the world chess champion, Garry Kasparov, in 1997. In fact, two decades before Kasparov's sensational defeat, a chess program using straightforward exhaustive search had already won the Minnesota Open Chess Championship in 1976 [1]. But, as Berliner predicted in 1978, "even if a full width search program were to become World Chess Champion, such an approach cannot possibly work for Go and this game may

¹A few basic knowledge about the game may be helpful but not essentially needed to appreciate the present paper. We shall skip all details about the game due to the space constraints. As the West is gradually attracted to this fascinating oriental game, there are quite a lot of good introductory books available in English; for example, see [8], or simply type "Go Game" on Amazon's search engine.

have to replace chess as the task par excellence for Artificial Intelligence" [1, 16].

Putting aside the extremely complicated overall strategy of the game (most skillful players believe it's not computational at all), a simple analysis is enough to convince AI researchers that a classical search-based algorithm doesn't work for computer Go. The size of the standard Go board is 19×19 , which is almost six times larger than the chess board. Moreover, the rule is extremely simple. A player can enter a stone almost everywhere with all kinds of purposes. In other words, the number of available moves in average is about 200 (only about 30 moves available in a typical chess game). As a result, the game-tree with only 4 moves in depth contains about 1.6 billion nodes. Nevertheless, computer Go designers never completely give up the look-ahead technique. For example, the well-known α - β tree is used in a recent investigation [15], where the size of the board is reduced to 5×5 . The problem is, when the size of the playing board is increased, the "optimized" result in an isolated 5×5 territory cannot be extended beyond the border; on the contrary, we may just have an opposite result in most nontrivial cases.

Albeit the fact that human brain is not good at mass computation, look-ahead technique is still used by most human Go players; for advanced players, they can check as many as 30 moves ahead [5]. It is clear that such human look-ahead approach must be highly focused, well planned, and goal-directed. With proper conceptual knowledge and goal-directed planning, the game tree can be subtly pruned to a feasible size. Lehner's Contingency Plan Goal Tree (CPGT) [10] is a forerunner of such goal-directed planning for computer Go. Another commonly used approach is pattern recognition, where most typical patterns are stored and recognized during the game. This idea can be traced back to Zobrist's dissertation [18] in 1970. Cazenave's Gogol [3] is a more recent implementation using the pattern recognition approach. Although computer Go is still in its infancy in terms of the strength, the journey of attempts is rich. There are many other new approaches suggested or implemented for computer Go programs in the past decade, e.g., neu-

ral network approach, cognitive model, machine learning, fuzzy logic, and so on. Due to the space constraints, we shall omit them and refer the interested reader to [2, 11] for more recent surveys. Also, see [14] for some important computer Go programs between 70's and late 80's which are of historical interest.

Our Contribution: The Canonical Sequence Directed Tactics Analyzer (CSDTA, hereafter) should be considered as a tool to be integrated in a complete computer Go program. The analysis of CSDTA is based on a rich collection of optimized sequences of moves called canonical sequences (Joseki, literally means standard deployment). While recognizing some predefined patterns to guide the strategic goal or tactical decision during the game is a common approach in computer Go programs, no existent Go program to our knowledge has built a collection of canonical sequences in the program as rich as CSDTA. Also, in order to use the sequences in a more flexible way, we propose an evaluation function to define the similarity between the current game and canonical sequences. The original idea of using canonical sequences was proposed by Koh in [9].

2. CSDTA

The underlying idea of CSDTA is that: it recognizes the configuration on the playing board and consults the experiences of expert Go players to make up its decision (next computer move) through some evaluation matrices. The experiences of expert Go players are represented in form of canonical sequences. A canonical sequence is a sequence of local optimal moves with influence that may extend to remote territory. Each canonical sequence has been thoroughly evaluated by expert players for years, some of them have been considered as classic plays for centuries, and each has proven to be optimal for both players. On the other hand, if one player plays out of the sequence by a mistake or for other trade-off reasons, the opponent will be significantly benefitted if he/she can response accordingly. Our database contains 1,278 canonical sequences and their deviations based on the collections in Ishida's [7] and Ha's [5]. The deviations are beneficial to the player who play black stones if the opponent plays out of the canonical sequences. Traditionally, canonical sequences are classified into different categories according to their first moves. Each category carries a different agenda for further deployment. We further classify each category in our database according to their second moves as shown in Table 1. The second move of a canonical sequence determines how the opponent engages. Since a canonical sequence can be easily rotated, symmetrically transformed, and switched stone colors, we store only one sequence for all its trivial variations, and black stone always makes the first move.

CSDTA is not a complete Go game program with following restrictions: 1. It only plays the black stone; 2. It considers only one quadrant of the board; 3. It plays only from the deployment stage (opening-game) through the first half of contact-fighting stage (mid-game). Moreover, CSDTA does not consider either the whole board strategies or the strategies behind each canonical sequence. In a real game, the player determines which canonical sequence to be initiated in a quadrant based on his/her overall deployment strategy. Since CSDTA does not deal with strategy decision, it does not decide which canonical sequence to be used. We simply let the user (or a complete computer Go game program that employs CSDTA) initiate the first move.

No look-ahead algorithm is used in CSDTA, although we believe that including a light look-ahead search could significantly improve the decision quality. Also note that CSDTA is not a miniature of a standard Go game. A quadrant of the standard Go board is not identical to a 10×10 Go board. A 10×10 board is an enclosed world having four corners and four edges, while a quadrant considered by a canonical sequence is a corner and the two open sides opposite to it. More importantly, a corner play usually involves intensive contact fighting with some tricky moves that can't be found in standard canonical sequences. Another kind of patterns for recognizing urgent moves (*Kyuba*), vital moves (*Kyusho*), or tactical moves (*Tesuji*) are needed for this purpose.

3. Evaluation Matrices

The main difference between the evaluation method of CSDTA and most classical pattern recognition approaches used in a computer Go program is that, the sequence to be recognized in CSDTA is a temporal patterns, i.e., the developing order of each canonical sequence is essential. CSDTA should shift the focus according to the game being played and try to form the black stones into a shape identical or similar to the predefined sequences. When CSDTA cannot find an exact match, some similar canonical sequences will be selected and then CSDTA will further weight the selected canonical sequences and find the best one for next move. There are two stages in CSDTA. The first stage defines similar canonical sequences and the second stage decides the next move. Each stage uses a different evaluation matrix called importance-matrix. A quadrant of each matrix is shown in Figure 1. A complete matrix can be obtained by simply rotating the quadrant to the rest three quadrants. We will use M_1 and M_2 to denote the two matrices for the first and second stages, respectively.

Stage One – Selection of Similar Canonical Sequences:

We first define the degree of difference for each canonical sequence stored in the database with respect to the current

2^{nd} moves	1^{st} moves (Total:1278)						
	3-3	3-4	4-4	5-3	5-4	6-3	6-4
3-3	-	1	3	33	5	1	-
3-4	-	-	2	193	117	1	2
3-6	-	-	298	-	-	-	-
4-4	10	-	-	-	-	-	-
4-5	2	-	-	53	-	-	-
4-6	4	1	6	-	-	-	-
5-3	-	267	-	-	-	-	-
5-4	-	182	-	-	-	-	-
6-3	-	25	-	-	-	-	-
6-4	3	36	-	-	-	-	-
Others	1	2	30	-	-	-	-
Total	20	514	339	279	122	2	2

Table 1. Collection of Canonical Sequences and Deviations

game. Following the simple philosophy that we should pay much more attention to the opponent's last move, we simply put the focus on the last move in stage one. Although CSDTA is designed to play black stones only, it is possible that the human player would skip his/her turn (in a real game, the player would have shifted his/her attention to another quadrant) and hence the last move may not always be a white move. For each canonical sequence in the database, we extract an initial segment up to the focus (the last move of the game). Note that the number of moves in the initial segment of a concerned canonical sequence may not be the same as the current game.

For convenience, let Ω denote the current game, F the focus, and ω the initial segment of a concerned canonical sequence up to F . Consider the game played by CSDTA against a human player (white stone) in Figure 2. We shall explain how CSDTA enters 9_B in response to 8_W . We have

$$\Omega = \{(G, 4)_B, (H, 6)_W, (D, 3)_B, (F, 6)_W, (E, 5)_B, (H, 4)_W, (H, 3)_B, (I, 4)_W\}$$

and $F = (I, 4)$. Here we consider the evaluation of canonical sequences (A) in Figure 2 and have

$$\omega = \{(G, 4)_B, (H, 6)_W, (D, 3)_B, (H, 3)_W, (H, 4)_B, (I, 4)_W\}.$$

The calculation of the difference degree is rather straightforward: Find the mismatch points between Ω and ω and sum up the values of M_1 at the mismatch points with the focus shifted to F . In this example, the mismatch points are $(F, 6)_W$, $(E, 5)_B$, $(H, 3)_W$, and $(H, 4)_B$. As the focus landed at $(I, 4)$, their corresponding values are 2, 3, 7, and 8. Thus, the degree of difference is 20. Similarly, one can find the degrees of difference for canonical sequence (B), (C), and (D) in Figure 2, which are 22, 26, and 26, respectively. All shaded stones in Figure 2 are moves after $(I, 4)$,

the focus, which are not yet played and hence will not be considered.

CSDTA computes the difference degree for every sequence in our database with respect to the current game. Let d be the minimum difference degree. All canonical sequences with degree of difference less than or equal to $d \times (4/3)$ are called *similar canonical sequences* with respect to the current game. Every similar canonical sequence will be passed to stage two for further evaluation. The ratio, $4/3$, is an experimental value, which is obtained by observing the performance of CSDTA and $4/3$ provides better results based on our collection.

As our example shown in Figure 2, the canonical sequence (A) in Figure 2 is the one with the minimum degree of difference, which is 20. Therefore, all canonical sequences with degree of difference less than or equal to 26 will be further evaluated in the next stage.

Stage Two – Selection of the Next Move: Although the canonical sequences selected in stage one are considered similar, their next moves may be very different. Since the vicinity of a potential move in the current board is critical in determining the goodness of the move, it is natural to put our focus on the move for evaluation, while the last move is still important to us. Therefore, CSDTA uses two focuses in stage two: the last move of the game and the next move of the canonical sequence being evaluated.

Since the next move is much more sensitive to the surrounding than the last move that has been entered, we use a different matrix in which the weight is reduced rapidly when the points are moving away from the focus (see Figure 1). The pattern is similar to Sander-Davies's influence pattern [12] and Yen and Hsu's Rough Inference [17]. Moreover, we think the type of mismatch at each point should be considered. We therefore adjust the weight according to the type of mismatch as shown in Table 2, where $f(w)$ is defined and w is the entry of the importance-matrix, M_2 .

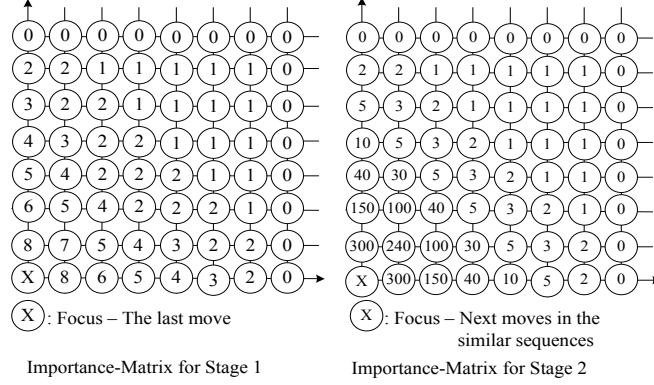


Figure 1. Importance-Matrices, M_1 and M_2

Canonical Sequence	Board (Current Game)		
	Empty	White	Black
Empty	0	$2w$	$f(w)$
White	$w/2$	0	$w/2$
Black	w	$2w$	0

$$f(w) = \begin{cases} 2w & \text{if the black stone reduces} \\ & \text{the liberties of the cluster;} \\ w & \text{otherwise.} \end{cases} \quad (1)$$

Table 2. Weight Adjustment in Stage Two

The final weight of each selected canonical sequence is simply 10,000 subtracting the sum of adjusted weights in M_2 as discussed above. Figure 2 shows four similar canonical sequences and their final weights. Consider sequence (A) again. For focus one, the weights corresponding to the four mismatch points are 240, 300, 5, and 5. After adjustment, we have $240/2 + 300 \times 2 + 5 + 5 \times 2 = 735$. For focus two, we have 300, 240, 40, and 30, and after adjustment, we have $300/2 + 240 \times 2 + 40 + 30 \times 2 = 730$. Therefore, the final weight of canonical sequence (A) in Figure 2 is $10,000 - 735 - 730 = 8,535$. Similarly, the final weight of canonical sequence (D) is 9,406, which turns out to be the one with highest weight among all similar canonical sequences, and hence its next move 11_B at (I,3) becomes CSDTA's next move 9_B . Note that in (1) we need to find the number of liberties of a cluster. Thus, CSDTA needs to maintain every cluster on the board, which is the only unit higher than a single stone to be recognized by CSDTA. Each cluster is maintained by two linked-lists for connected stones in the same color and their liberties, respectively. It is clear that the cluster is a necessary unit for further developing but not essential in the present paper.

4. Assessment

Memory requirement and response time of CSDTA is not an issue to today's PC standard. Our program use 150 KB to load all 1278 sequences into main memory and the response time for each move is negligible. Since CSDTA is not a

complete Go game program and no other assistant evaluation is used, it is not appropriate to appraise its strength. For further speculation, however, we shall give it a rough assessment in this section.

The Good: Compared to KYU in [12] which is not a complete program either but rated about 30 kyu in the opening game, and it can play about 20 acceptable but, as the authors mentioned, "conservative and unimaginative" moves in the whole board (5 or 6 per quadrant), CSDTA plays much better than that. In general, CSDTA can respond with an excellent move if the selected similar canonical sequence has a degree of difference less than 20, and it can play fairly well when the degree is between 20 and 30. When the degree is between 30 and 40, CSDTA in most cases can still play acceptably. Only when the degree of difference exceeds 40, the quality become unpredictable. In the game shown in Figure 2, the winner, canonical sequence (D), has 26 as the degrees of difference; CSDTA suggests (I,3) as the ninth move of the game. In this example, a good shape for black has successfully deployed.

In most cases, if the minimum degree of difference is less than 30, there always exist canonical sequences with weights higher than 9,000. This indicates that CSDTA is not likely to suggest a bad move in a normal opening game. If the human player does not intentionally play against common canonical sequences, the degrees of difference of the selected similar canonical sequence will remain under 40 for about 15 moves. In a quadrant, 15 moves would have

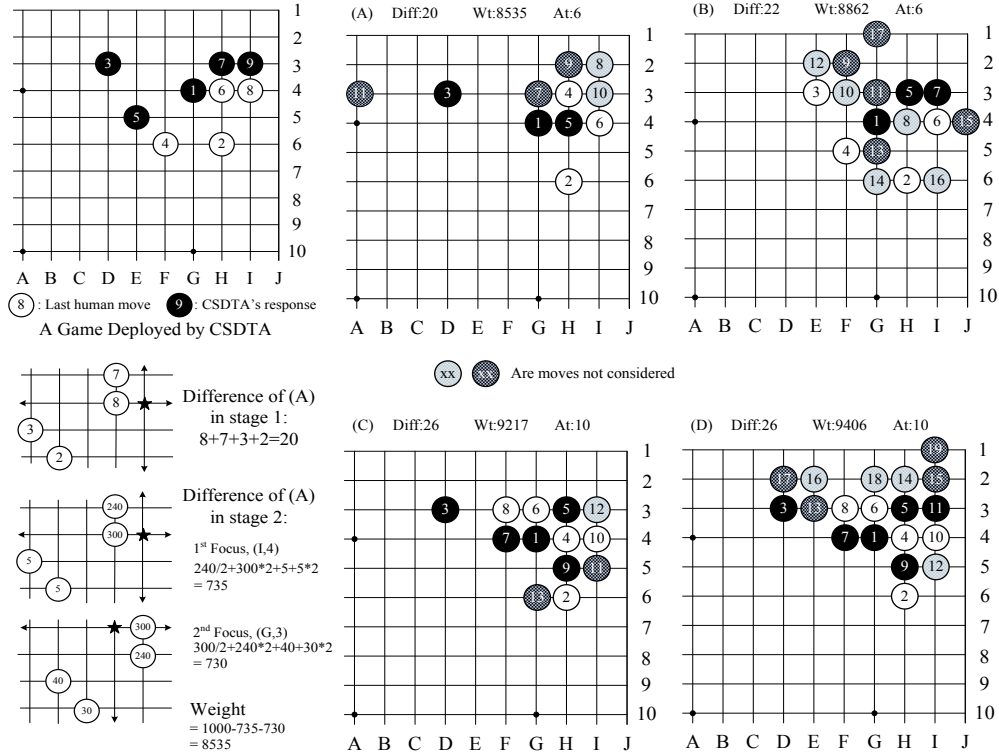


Figure 2. A Game Against CSDTA and Four Canonical Sequences Evaluated by CSDTA

taken the game into the mid-game stage. Extending to a complete board, CSDTA can play about 60 good moves. We consider to have 60 good moves without too much cost a significant achievement.

The Bad: CSDTA mechanically select the “best” canonical sequence without considering the strategy behind the sequence. Also, as no other technique used in CSDTA, it is not surprised that CSDTA may be able to prepare a good tactics deployment for a battle, but it cannot finish the battle and win a territory. Another major shortcoming of CSDTA is that it cannot handle some obvious abnormal moves. On the other hand, we do not really consider these real disadvantages of CSDTA. It should be easy to set a cutting line according to the difference degree and weight for CSDTA to quit and let another method take over. In fact, many situations in which CSDTA handles badly by their nature should be solved by using another approaches, e.g., search-based algorithms or another kind of patterns as we mentioned earlier at the end of Section 2.

5 Future Study and Conclusion

5.1 Future Study

In this subsection we point out four possible directions for future development.

Overall Game Planning: For a human player, a canonical sequence is selected according to his/her overall deployment strategy. In other words, each canonical sequence carries a strategy behind. Thus, it is more appropriate if the program can select canonical sequences according to its overall plan. One possible way is to associate a strategy to each canonical sequence in our database. Such strategy can be something like, for example, “this sequence attaches importance to the corner territory” or “this sequence trades the corner territory for influence in the center”. That information will become a built-in strategy for each canonical sequence. Once the goal is determined, we should weight the canonical sequences according to the goal.

Critical Patterns: The knowledge base of CSDTA contains only canonical sequences (Joseki). If the minimum degree of difference becomes large, CSDTA will still try to extract a similar part from *not-so-similar* canonical sequences to determine the next move. As a result, the quality becomes unpredictable. When the game varies from the standard sequence, it is more appropriate to consult a different kind of patterns known as “Critical Patterns”. In general, such kind of patterns provide urgent moves (*Kyuba*), vital moves (*Kyusho*), and tactical moves (*Tesuji*), which are not a sequence of moves but a few critical points the player

needs to occupy whenever a certain pattern is recognized. In particular, critical patterns should suggest how to make two eyes, to kill an intervening enemy, to escape stones in danger, to connect with friendly groups, etc. Adding some critical patterns can certainly extend the number of acceptable moves significantly.

Refining Importance-Matrices: Our importance-matrices are naive in the sense that the degree of importance is radiating uniformly from a single point (the focus). In human player's conception, this should not be the case. The formation of friend and enemy stones strongly affects the pattern of the radiation. Thus, to accurate our evaluation, we may adopt the idea of Yen and Hsu's Rough Inference [17] in designing the importance-matrices according to the shape on the playing board. Also, instead of using the same matrix, we should design two different matrices for the two focuses examined in stage two. Moreover, a different goal planned may direct CSDTA to select a different matrix.

Look-ahead Technique: Another direction for a promising improvement is to adopt a look-ahead program in CS-DTA in a few places. First of all, we can replace our adjustment function f (see Table 2) used in stage two, which is obviously inadequate, by an actual search-based/look-ahead program. For example, MIGOS (MIni GO Solver) presented in [15] is such a program using the well-known α - β tree to evaluate a 5×5 board. CSDTA can use MIGOS to examine the 5×5 vicinity around the focuses. Also, when the minimum degree of difference is higher than a constant (for example, 40), CSDTA should halt and use a look-ahead program to finish the battle or to actually occupy some territory. In such a way, abnormal moves can also be handled.

5.2. Conclusion

Since the Go game is so sophisticated, it is not surprising that no single approach can build a satisfactory computer Go program. Our investigation does not intend to build a complete Go program. Instead, our goal is to provide a feasible approach in helping improve computer Go programs by using canonical sequences. The notion of similar canonical sequences and weighting matrices make the use of canonical sequences more flexible than just recognizing predefined patterns. Simply put, it is worthwhile to obtain 10 excellent moves for tactical deployment in a quadrant at little cost of computing time. If CSDTA cooperates with an overall strategic analyzer, stage determining program, focus shifter, and global and local evaluator, it can easily play more than 40 excellent moves on a whole board in the opening game, and this is about the same outcome a human player expects to achieve by way of studying canonical sequences. We believe that CSDTA has achieved its goal.

References

- [1] H. J. Berliner. A chronology of computer chess and its literature. *Artificial Intelligence*, (10):201–214, 1978.
- [2] Bruno Bouzy and Tristan Cazenave. Computer go: An AI oriented survey. *Artificial Intelligence*, 132(1):39–103, 2001.
- [3] Tristan Cazenave. Automatic acquisition of tactical Go rules. In H. Matsubara, editor, *Proceedings of the 3rd Game Programming Workshop*, pages 10–19, Hakone, Japan, 1996.
- [4] H. S. Green. Go and artificial intelligence. *Computer Game Playing: Theory and Practice*, pages 141–51, March 1985.
- [5] Chan-Suk Ha. *Fundamentals of Protocols in Playing Go Games*, volume I and II. The Korean Go Association, Seoul, Korea, 1988.
- [6] Feng-hsiung Hsu. IBM's Deep Blue chess grandmaster chips. *IEEE Micro*, 19(2):70–81, March 1999.
- [7] Masao Ishida. *Canonical Sequence Dictionary*. World Culture Publishing, Taipei, Taiwan, 1984.
- [8] Janice Kim. *Learn to Play Go: A Master's Guide to the Ultimate Game*. Ishi Press International, April 1995.
- [9] Hikyoo Koh. A model for go player by stage-based strategies and tactics. In *The Conference on Modelling and Simulation*, Univ. of Pittsburgh, Pittsburgh, 1990.
- [10] Paul Edward Lehner. Planning in adversity: A computational model of strategic planning in the game of go. University of Michigan, 1981. Ph.D dissertation.
- [11] M. Mueller. Computer go: A research agenda. In *Lecture Notes in Computer Science*, number 1558, pages 252–264. Springer Verlag, Heidelberg, Germany, 1998.
- [12] Peter T. Sander and D. Julian M. Davies. A strategic approach to the game of go. In *In Computer Game Playing: Theory and Practice*, pages 167–76. Ellis Horwood, New York, 1985.
- [13] Clause Shannon. Programming a computer for playing chess. *Philosophical Magazine*, 41:256–275, 1950.
- [14] David Stoutamire. Machine learning, game play, and Go, chapter 2: Computer gamesmanship and Go. TR-91-128, Case Western Reserve University, 1991.
- [15] E. van der Werf, H. van den Herik, and J. Uiterwijk. Solving go on small boards. *ICGA Journal*, 26(2):92–103, June 2003.
- [16] L. J. Yedwab. On playing well in a sum of games. MIT/LCS/TR-348, MIT Laboratory for Computer Science, Cambridge, MA., 1985. M.S.. Thesis.
- [17] Shi-Jim Yen and Shun-Chin Hsu. A positional judgment system for computer go. *Advances in Computer Games*, 9:313–326, 2001. Universiteit Maastricht.
- [18] A. L. Zobrist. Feature extraction and representation for pattern recognition and the game of go. University of Wisconsin, 1970. Ph.D. dissertation.