Publication VI

Esa A. Seuranen. 2009. Entropy in go. ICGA Journal, volume 32, number 1, pages 34-40.

ENTROPY IN GO

Esa A. Seuranen[1]

Helsinki, Finland

ABSTRACT

In this note the Shannon entropy is used to measure complexity and performance in the context of (computer) Go. To support the use of entropy some experimental results are presented.

1. INTRODUCTION

Go is at least 2500 years old. It is a complete-information board game for two players. The rules of the game are quite straightforward. The play begins from an empty 19x19 board and players place a stone of their own colour (or pass) on the board in turn. A stone can be placed on any vacant location as long as the placement does not repeat an earlier board situation. Orthogonally adjacent stones of the same colour belong to the same block. A block has as many liberties as it has vacant locations orthogonally adjacent to any stones in the block. If a player's placement takes the last liberty of a block, the block is captured (removed from the board). If blocks of both players are reduced to zero liberties, only the opponent's blocks are captured. The game ends when both players pass, after which dead (the opposing player can capture regardless of the defending player's actions) stones are removed and the player with a larger *score*, i.e., the sum of stones and territory (the vacant locations enclosed by the player's stones) on the board wins. For detailed information on computer Go the reader is referred to Bouzy and Cazenave (2001), Chen (2001), Müller (2002), van der Werf (2004), Seuranen (2007a), and their references.

The Shannon entropy is a measurement of uncertainty for a random variable, consider, e.g., the question: how many bits are needed on average to describe the values of a random variable? In other words, entropy represents a limit for lossless compression. It can be argued that there is a connection between good compression and artificial intelligence[2].

In this note we investigate how entropy can be used to measure complexity and performance. We take an approach similar to Shannon (1951), in which printed English is studied. The course of the note is as follows. Section 2 contains the relevant definitions. In Section 3 we present some experimental results to support the use of entropy. Section 4 contains a discussion and a conclusion.

2. DEFINITIONS

A thorough treatment on entropy can be found in several text books, for instance, in Young (1971), but here we will need only the basic definition. The entropy H of a random variable X with a probability mass function $p(x)$ is defined by

$$H(X) = -\sum_{x \in X} p(x) \ln p(x).$$

The entropy is at maximum for the uniform probability distribution and at minimum when all the probability is concentrated in a single value (i.e., there is no uncertainty of the next value). Entropy is not a new concept

[1]Department of Communications and Networking, Helsinki University of Technology, Finland, email: esa.seuranen@tkk.fi

[2]The Hutter Prize is motivated by the concept connection (`http://prize.hutter1.net`). There has also been some discussion on taking the same approach to Go (`http://senseis.xmp.net/?CuttingTheGordianKnot`).

to (computer) Go, as entropy has been used to measure differences between distributions Araki *et al.* (2007), Chaslot *et al.* (2008). Here we would like to investigate the average entropy per move, but naturally this kind of value depends on both the observer (predictor) and the observed games. If we know the predictor sufficiently well, i.e., we known $p(x)$, we can calculate the entropy directly. However, such information is not always available.

Definition (predictor): A *predictor* is a system which converts a game record deterministically into a sequence of integers.

The i-th integer in the sequence represents the number of predictions which the predictor needs to guess correctly the i-th move in the game record. We note that by the used predictor it is possible to reconstruct the original game record from the sequence. In this note, we use the notation $H_{\mathcal{P}}^{\mathcal{G}}$ for the entropy in game records $\mathcal{G}$ for the predictor $\mathcal{P}$. Let $q_j(\mathcal{G},\mathcal{P})$ be the frequency of j in the sequences generated by predictor $\mathcal{P}$ from the set of game records $\mathcal{G}$. Then we define that

$$H_{\mathcal{P}}^{\mathcal{G}} = -\sum_j q_j(\mathcal{G},\mathcal{P}) \ln q_j(\mathcal{G},\mathcal{P}).$$

In our investigations, we use the following six predictors, $\mathcal{P}$.

`Random:`	The move is predicted (pseudo)randomly (with a given seed for the random number generator) from legal moves using a uniform probability distribution.
`Nearby:`	The nearest (Manhattan distance) legal move to the previous move is predicted, starting from the location above and proceeding into the counter-clockwise direction.
`CP:`	CrazyPatterns (`http://remi.coulom.free.fr/Amsterdam2007`).
`GNU-lvl:`	GNU GO (`http://www.gnu.org/software/gnugo`) 3.7.10, where `lvl` is the level parameter for GNU GO. GNU GO provides a list of ten moves.
`MoGo-sims:`	MOGO (`http://lri.fr/~gelly/MoGo.htm`) release 1, where `sims` is the number of simulations in thousands. MOGO provides a list of five moves.
`Perfect:`	A theoretical player/predictor, which achieves the best possible score (regardless whether it is a win or loss) from the current position assuming that the opponent is also a perfect player.

We note that by the above definition a perfect player maximizes the score whereas the current state-of-the-art computer-Go players (including MOGO) maximize the probability of winning. One should also note that such players may do especially badly in predicting moves in clearly decided games.

As some predictors provide only a partial list of legal moves, we calculate an estimate of entropy using the top five predictions and "distributing prediction misses uniformly over the other legal moves", i.e.:

$$\hat{\mathcal{H}}_{\mathcal{P}}^{\mathcal{G}} = -q_{\text{miss}}(\mathcal{G},\mathcal{P}) \ln \frac{q_{\text{miss}}(\mathcal{G},\mathcal{P})}{s} - \sum_{j=1}^{5} q_j(\mathcal{G},\mathcal{P}) \ln q_j(\mathcal{G},\mathcal{P})$$

where $q_{\text{miss}}(\mathcal{G},\mathcal{P}) = \sum_{j>5} q_j(\mathcal{G},\mathcal{P})$ and s is a branching factor so that $H_{\texttt{Random}}^{\mathcal{G}} \approx \ln s$.

3. EXPERIMENTS

Studying moves in high quality games (that is, games between professionals or high-level amateur players) has been quite popular (e.g., see van der Werf *et al.* (2002), Coulom (2007), Bouzy and Chaslot (2005), Stern, Herbrich, and Graepel (2006). The aim of these studies has naturally been to achieve a high prediction rate (the next move is predicted correctly) as possible. The best[3] published results are 37% (Stern, 2008). It should be noted, that a high prediction rate alone does not result in a strong player, as these specialized prediction systems tend to make game-losing blunders in tactical situations - - even though otherwise they seem to play rather well.

In contrast, published questionnaire studies with human players appear to be rather limited. In Althöfer and Snatzke (2003) a human operated as a decision maker choosing a move from the ones proposed by computer players - - with the result that even a small pool of move candidates is sufficient to avoid the biggest blunders; in van der Werf *et al.* (2002) humans were asked to choose between a real game move and a close by random move yielding a correlation between the players' rank and their prediction rate.

[3]STEENVRETER's next move prediction performance is around 38%. Maximizing this number instead of minimizing (effectively) the average ranking of the next move (van der Werf *et al.*, 2002) rates well beyond 40%. Fourty per cent should be obtainable (Personal communication with Erik van der Werf, 2008).

$\mathcal{P} \setminus \mathcal{G}$		11 kyu	8 kyu	5 kyu	2 kyu	2 dan	5 dan	Pro
Nearby	$H_{\mathcal{P}}^{\mathcal{G}}$	3.78	3.90	3.91	3.97	4.07	4.13	4.39
	Top-1	15.0	14.2	13.3	13.4	12.5	12.2	11.3
	Top-5	46.2	43.9	42.8	41.7	39.1	37.8	34.4
	Average ranking	25.8	28.5	27.6	28.9	30.9	32.2	42.3
CP	$H_{\mathcal{P}}^{\mathcal{G}}$	2.59	2.64	2.61	2.57	2.66	2.74	3.02
	Top-1	40.1	38.9	39.4	40.4	37.7	36.6	33.3
	Top-5	72.2	70.8	71.1	71.5	69.6	67.8	61.1
	Average ranking	8.92	9.2	8.52	8.06	8.5	9.28	11.9
	"Mean log-evidence"	-2.46	-2.51	-2.47	-2.42	-2.53	-2.61	-2.88

Table 1: Results with `Nearby` and `CP`.

$\mathcal{P} \setminus \mathcal{G}$	11 kyu	8 kyu	5 kyu	2 kyu	2 dan	5 dan	Pro
Nearby	4.45	4.54	4.59	4.63	4.73	4.78	4.9
CP	3.06	3.15	3.13	3.1	3.23	3.32	3.66
GNU-1	4.98	4.78	4.67	4.56	4.57	4.6	4.6
GNU-5	4.99	4.81	4.67	4.58	4.59	4.58	4.58
GNU-10	4.93	4.73	4.58	4.47	4.43	4.44	4.48
MoGo-1	4.46	4.51	4.51	4.48	4.46	4.46	4.63
MoGo-10	4.96	4.94	4.84	4.83	4.72	4.7	4.81
MoGo-20	5.05	5.01	4.92	4.9	4.76	4.75	4.85

Table 2: Results regarding $\hat{\mathcal{H}}_{\mathcal{P}}^{\mathcal{G}}$.

3.1 Computational Experiments

The average length and average number of legal moves are 218 and 250, respectively, based on 1334 professional even game records from Andries Brouwer's collection (`http://homepages.cwi.nl/~aeb/go/games`). We use the alphabetically first 200 game records from these games along with 200 even game records between 11 kyu, 8 kyu, 5 kyu, 2 kyu, 2 dan, and 5 dan players[4] from KGS Go Server (`http://www.gokgs.com`). The KGS game records are from September 2007 by players chosen more or less arbitrarily.

In Table 1 the entropies of the predictors `Nearby` and `CP` are calculated; the cumulative percentages of the next move being in top-k predictions are widely used in the literature - for comparison such values for $k = 1$ and $k = 5$ are displayed as well; also the average ranking of the next move is shown, as it (or more exactly, the error in ranking or its square) is mentioned in the literature quite often; and finally, "mean-log evidence"[5] (mean logarithm of the probability of the next move) is calculated for `CP`. Furthermore, the branching factor $s = 299$ was obtained, i.e., $H_{\texttt{Random}}^{\mathcal{G}} \approx 5.7$ regardless of $\mathcal{G}$.

The calculated values of $\hat{\mathcal{H}}_{\mathcal{P}}^{\mathcal{G}}$ are displayed in Table 2. The rather large difference between $\mathcal{H}$ and $\hat{\mathcal{H}}$ straightforwardly follows from the "uniform distribution of prediction misses". From these results one can notice, that GNU GO's prediction capability seems to increase along `lvl`. Interestingly, the inverse seems to be true for MoGo-`sims` - - apparently MoGo starts with a quite traditional probability distribution for legal moves. As the amount of simulations is increased, MoGo finds its "own" playing style (thus reducing its ability to predict games played in a "traditional" style). Surprisingly, `Nearby` fares better in predicting lower level games than GNU GO and MoGo. The specialized prediction system `CP` has a performance on its own level. Remarkably, `CP` does clearly better at predicting lower/middle level games than at high/professional level games.

A similar study on 240 professional 9x9 games from GoBase (`http://gobase.org/9x9`) was also conducted. We shall refer to this game collection by `9x9`. In these games the average length is approximately 47 moves, there are on average 57 legal moves in a position and the branching factor $s = 68$ was obtained. The results are summarized in Table 3, showing MoGo becoming better at predicting as simulations are increased (in contrast to the 19x19 results).

[4]The amateur ranks in Go start from 30 kyu and decrease all the way to 1 kyu. Thereafter dan levels in increasing order are used up to 7 dan, which is considered the be equivalent to the first professional rank. An average tournament player in Europe has a rank around 7 kyu.

[5]The probability distribution for the next move was recalculated from the probability colour map (as apparently `CP` does not provide a numerical probability distribution directly); hence we used quotes.

$\mathcal{P}$	`Random`	`Nearby`	`GNU-1`	`GNU-10`	`MoGo-1`	`MoGo-20`	`MoGo-50`	`MoGo-100`
$H^{9x9}_{\mathcal{P}}$	4.22	3.54	-	-	-	-	-	-
$\hat{H}^{9x9}_{\mathcal{P}}$	4.22	3.79	2.99	2.87	3.05	2.77	2.72	2.68

Table 3: Results with professional 9x9 games.

$\mathcal{P}$	all	–1k	2k–	–1d	1k–4k	5k–	–3d	2d–1k	2k–4k	5k–
	(50)	(24)	(26)	(17)	(18)	(15)	(11)	(13)	(11)	(15)
1.	8.9	10.8	7.2	11.8	7.0	8.0	11.4	11.2	6.5	8.0
2.	7.7	10.8	4.9	11.4	7.8	3.6	13.2	9.7	7.1	3.6
3.	5.9	6.1	5.6	5.9	6.3	5.3	6.5	6.6	6.5	5.3
4.	4.9	4.4	5.4	5.5	6.3	2.7	5.9	4.0	9.5	2.7
5.	2.5	2.8	2.3	2.4	3.0	2.2	2.9	3.5	2.9	2.2
miss	78.2	65.1	74.6	63.0	69.6	78.2	60.1	65.0	67.5	78.2
$\hat{H}^{\mathtt{EGC}}_{\mathcal{P}}$	5.05	4.87	5.2	4.79	5.05	5.3	4.68	4.87	4.97	5.3
moves	57	36.4	45.9	29.9	35.1	36.1	23.2	27.9	27.3	36.1

$\mathcal{P}$	–4d	3d–1d	1k–2k	3k–5k	6k–	`GNU`	`MoGo`	`CP`
	(7)	(10)	(11)	(12)	(10)			
1.	16.6	9.1	7.7	7.7	8.5			
2.	12.8	11.1	10.8	5.5	2.5			
3.	6.2	6.5	5.9	7.1	5.8			
4.	6.2	5.8	5.9	7.1	1.8			
5.	2.4	3.1	2.9	4.3	1.8			
miss	55.8	64.4	66.8	68.3	79.6	66.7–80	73.3–86.7	93.3
$\hat{H}^{\mathtt{EGC}}_{\mathcal{P}}$	4.49	4.86	4.94	5.01	5.32			
moves	17.1	24	26.8	25.7	28.6			

Table 4: The EGC2007 experiment results.

3.2 Humans as Predictors

Below we describe results from a questionnaire which took place at European Go Congress 2007. In the questionnaire the participants were asked to give 5 predictions for the next move in 15 game situations from professional games. We shall refer to this game-situation set by `EGC`. The game situations were chosen so that the next move was not trivial (at least not for the author, at that time 3 dan himself). Along the way the questionnaire was carried out we saw that each participant used in total 20 to 35 minutes to give all their predictions. The game situations and some summary over the proposed moves can be found in Seuranen (2007b).

There were 61 participants, but here we will regard only participants who gave at least 64 predictions (some participants gave consistently only few predictions per game situation). This leaves us with 50 participants. We analyzed the predictions of these players a few times, by dividing them first into two groups, then into three, four, and finally five groups, based on their rank.

The results of the experiment are summarized in Table 4. The player group is mentioned in the top row (e.g., --3d refers to players with rank 3 dan or better and 5k– to 5 kyu or less) and the number of players in the group is in parenthesis in the second row. The entries of rows 3 to 7 are the percentages, how well a player in the corresponding player group guessed the next move with the corresponding prediction. Similarly the row "miss" is the percentage of completely missing the next move. The last row is the average number of the proposed different moves per game position by the entire group.

For completeness (although 15 game positions are rather few for single entities) the miss percentages for GNU Go, MoGo and `CP` are also listed in Table 4. GNU Go seemed to fare better with higher `lvl` whereas MoGo's performance peaked around 10,000 simulations. Apparently the `CP`'s earlier good performance in Table 1 cumulates over the numerous easier (or even trivial) moves and the poorer performance on harder moves (such as the ones in `EGC`) does not deteriorate the overall performance that much. The "mean log-evidence" of `CP` is -5.18 for `EGC`, which supports this interpretation.

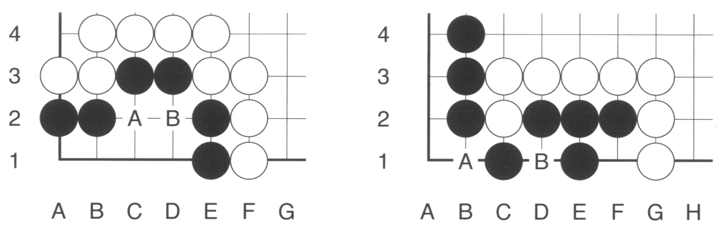

Figure 1: Moves of equal value are not necessarily equal.

4. DISCUSSION AND CONCLUSION

The discussion below is carried out along two dimensions. Section 4.1 deals with complexity. Section 4.2 concentrates on performance. In Section 4.3 a conclusion is provided.

4.1 Complexity

There are many ways to estimate complexity, but perhaps game-state size (number of different game positions) and game-tree size (minimum size of a search tree necessary to solve the game) are the most commonly used for games (Allis, 1994). The game-tree size is usually estimated as s^n, where s and n are the branching factor (often estimated by the average number of legal moves) and average game length, respectively. It is easy to see that $H_{\texttt{Random}}$ corresponds to the game-tree complexity, i.e.,

$$\ln(s^n) = n \ln s = nH_{\texttt{Random}}.$$

The game-tree complexity does not depend on the level of play. This leads to some peculiarities, as can be seen from Bouzy and Cazenave (2001): the game-tree complexity of 9x9 Go is smaller than 15x15 GoMoku, although in the former humans still outperform a computer where as the latter is already solved. Therefore, one might be more interested in the set of rational games instead of all possible games. Yet, the rationality of moves is in the eye of the beholder (predictor). It is interesting to note from the "moves" row of Table 4 that stronger players (as a group) seem to propose fewer moves for the next move.

Let us now consider a game played between two perfect players. If the observer knows only the rules, the average entropy per move for the observer is $H^{\texttt{Perfect}}_{\texttt{Random}} \approx \ln 299 \approx 5.7$. In contrast, if the observer is also a perfect player, then the average entropy per move can be very small, e.g., $H^{\texttt{Perfect}}_{\texttt{Perfect}} = 0$ if there is the unique perfect game.

The existence of the unique game is an interesting question and depends (to some degree) on the definition of a perfect player. In the left diagram of Figure 1 both black A and B prevent White from capturing all black stones, but A leaves White with two ko[6] threats and B with three. In the right diagram Black can save the four stones by playing A or B, but the latter gives White a larger ko threat at F1. A strong player would never play B in either of these positions, even if the choice would not affect the score of the game (and perhaps a perfect player should not either). As a final noteworthy aspect from the left diagram in Figure 1, after Black has played A White has two ko threats starting from either B1, C1, D1, or D2. Assuming Black ignores the first threat, then D1 and D2 are better (Black is left with no ko threats). And assuming Black answers the first threat and ignores the second, then it is best to begin with C1 or D2 (Black is left with no ko threats). As a conclusion, we may state that D2 is the best ko threat for White.

Finally, it can be stated that 5x5 Go has the unique game and 7x7 Go (Davies, 1994) is believed not to have the unique game (even when assuming that the symmetries in the beginning are discarded). Hence, the existence of the unique game for 19x19 Go is not likely. One could consider $H^{\texttt{Perfect}}_{\texttt{Perfect}}$ as a measurement of how complex

[6]If a player wants to play somewhere but is not allowed (because an earlier game position would be repeated), (s)he must play elsewhere. If the move is sufficiently threatening, the opponent answers it and then the player can play the move (s)he originally wanted. Such threatening moves are called ko threats, and the fight involving such moves is called a ko (fight).

or interesting Go is, but this is also not completely trouble-free, as some amount of the entropy follows from rather uninteresting choices (often the last few moves of a game are truly equivalent). Also when considering some other games (such as tic-tac-toe), solely using entropy for measuring complexity seems not so descriptive. Nevertheless, entropy seems to be a good way to measure (observed) complexity and can be used along with traditional measurements (concentrating on decision complexity), such as the size of the search tree or the size of the perfectly ordered search tree (Knuth and Moore, 1975).

4.2 Performance

As an example of a feature the predictor `Nearby` clearly demonstrates (Table 1) how the proximity of the previous move plays a decreasingly lesser role as the level of play increases. Measuring relevance of such features with entropy seems a valid option, as entropy perceives the accumulated probabilities and ignores rankings (if a feature or a predictor consistently accumulates probabilities, rearranging rankings into a more convenient order should not pose a problem).

The various ways of measuring the performance of a prediction system each seem to have its own perk: (1) cumulative probabilities of top-k moves are descriptive for small k, but they use only a fraction of the predictions; (2) average ranking is also quite descriptive and takes into account all predictions, although some slight inconsistencies can be observed from Table 1; (3) entropy as a value is not so descriptive, but it seems to provide the most consistent behaviour; and (4) mean-log evidence is roughly the same as entropy (with the need for a predictor to provide a probability distribution for the next move).

4.3 Conclusion

Based on the considerations above and the results shown in the tables, we may conclude that measuring the playing strength by predictions (whether using entropy or some other metric) seems less appealing. We may expect that a good player is a good predictor (as can be seen from Table 4), but the differences between players of similar playing strength are quite large (Seuranen, 2007b). Also, the inconsistencies between prediction performance and playing strength are quite apparent from Table 2. Interestingly `CP`, GNU GO and MOGO predicted the best games between 11 kyu (2 kyu), 2 dan, and 5 dan, respectively – ordering the predictors according to their playing strength – so perhaps it is possible to estimate playing strength by using game records of various levels.

5. ACKNOWLEDGEMENTS

This research was supported by the Nokia Foundation and the Academy of Finland under Grants No. 107493 and No. 110196. The author wishes to express his gratitude towards the referees of this note. They contributed considerably by shaping the definitions and the contents.

6. REFERENCES

Allis, L. V. (1994). *Searching for solutions in games and artificial intelligence.* Ph.D. thesis, Universiteit Maastricht, Maastricht, The Netherlands.

Althöfer, I. and Snatzke, R. G. (2003). Playing games with multiple choice systems. *Lecture Notes in Computer Science*, No. 2883, pp. 142–153.

Araki, N., Yoshida, K., Tsuruoka, Y., and Tsujii, J. (2007). Move prediction in Go with maximum entropy method. *Proceedings of the 2007 IEEE Symposium on Computer Intelligence and Games*, pp. 189–195.

Bouzy, B. and Cazenave, T. (2001). Computer Go: an AI-oriented survey. *Artificial Intelligence*, No. 132, pp. 39–103.

Bouzy, B. and Chaslot, G. (2005). Bayesian generation and integration of k-nearest-neighbor patterns for 19x19 Go. *IEEE 2005 Symposium on Computational Intelligence in Games* (eds. G. Kendall and S. Lucas), pp. 176–181, Colchester.

Chaslot, G. M. J. B., Winands, M. H. M., Szita, I., and Herik, H. J. van den (2008). Cross-Entropy for Monte-Carlo Tree Search. *ICGA Journal*, Vol. 31, No. 3, pp. 145–156.

Chen, K. (2001). Computer Go: knowledge, search, and move decision. *ICGA Journal*, Vol. 24, No. 4, pp. 203–215.

Coulom, R. (2007). Computing "Elo Ratings" of Move Patterns in the Game of Go. *ICGA Journal*, Vol. 30, No. 4, pp. 199–208.

Davies, J. (1994). 7x7 Go. *American Go Journal*, Vol. 29, No. 3, p. 11.

Knuth, D. E. and Moore, R. W. (1975). An analysis of Alpha-Beta pruning. *Artificial Intelligence*, Vol. 6, pp. 293–326.

Müller, M. (2002). Computer Go. *Artificial Intelligence*, No. 134, pp. 145–179.

Seuranen, E. A. (2007a). Introducing playing style to computer Go. *Proceedings of the Computer Games Workshop 2007 (CGW 2007)* (eds. H. J. van den Herik, J. W. H. M. Uiterwijk, M. H. M. Winands, and M. P. D. Schadd), MICC Technical Report Series 07-06, pp. 81–91, Maastricht University, Maastricht, The Netherlands.

Seuranen, E. A. (2007b). Results summary of "Side Event: Predict professional moves" from European Go Congress 2007. http://users.tkk.fi/eseurane/EGC2007.

Shannon, C. E. (1951). Prediction and entropy of printed English. *Bell System Technical Journal*, No. 30, pp. 50–64.

Stern, D., Herbrich, R., and Graepel, T. (2006). Bayesian pattern ranking for move prediction in the game of Go. *ICML '06: Proceedings of the 23rd international conference on Machine learning*, pp. 873–880, ACM Press, New York.

Stern, D. H. (2008). *Modelling uncertainty in the game of Go.* Ph.D. thesis, University of Cambridge, Cambridge, UK.

Werf, E. C. D. van der (2004). *AI techniques for the game of Go.* Ph.D. thesis, Universiteit Maastricht, Maastricht, The Netherlands. ISBN 90–5278–445–0.

Werf, E. C. D. van der, Uiterwijk, J. W. H. M., Postma, E. O., and Herik, H. J. van den (2002). Local move prediction in Go. *3rd International Conference on Computers and Games*, Edmonton.

Young, J. F. (1971). *Information Theory.* Butterworths, UK, London, UK.