

# History and Territory Heuristics for Monte Carlo go

Bruno Bouzy<sup>1</sup>

<sup>1</sup> Université Paris 5, UFR de mathématiques et d'informatique, C.R.I.P.5,  
45, rue des Saints-Pères 75270 Paris Cedex 06 France,  
tél: (33) (0)1 44 55 35 58, fax: (33) (0)1 44 55 35 35,  
email: bouzy@math-info.univ-paris5.fr,  
http: www.math-info.univ-paris5.fr/~bouzy/

## Abstract

This paper assesses two heuristics within the 19x19 Monte Carlo go framework of INDIGO [4, 5, 7]: the territory heuristic and the history heuristic, both in their "internal" and "external" versions. The external territory heuristic is more effective, leading to a 40-point improvement on 19x19 boards. The external history heuristic brings about a 10-point improvement. The internal territory heuristic yields a few points' improvement, and the internal history heuristic has already been assessed on 9x9 boards [7]. Most of these heuristics were used by INDIGO at the 2004 Computer Olympiad [10].

**Keywords:** Computer Go, Monte Carlo, Territory, History Heuristic.

## 1. Introduction

Monte Carlo (MC) approaches to game playing are commonly used in hidden information games such as Poker or Scrabble, or in game containing randomness such as Backgammon. In Poker, POKI developed by the GAMES group at the University of Alberta uses simulations [2]. In Scrabble, MAVEN, designed by Sheppard, uses such simulations [14, 15]. Rollout experiments are also used in Backgammon [17]. In go, after GOBBLE, the first 9x9 MC go program, by Brüggmann [8], various other 9x9 MC go programs have been created over the last years: VEGOS by Kaminski [11], OLEG by Helmstetter [7], GO81 by Raiko [12] and DUMBGO by Hamlen. Until 2002, INDIGO [4] remained a classical 19x19 go program based on local tree search [3], and on extensive knowledge [6]. Since then, it has associated knowledge with a Monte Carlo approach [5], and has used progressive pruning [2, 7]. Thus, to this extent, INDIGO can be considered as the first 19x19 MC go program. Figure 1 provides an overview of the two-step move decision process used in INDIGO in 2003. The pre-selection module gives  $N_{\text{select}}$  moves, or  $N_s$  for

short, to the MC module that, in turn, selects the best move among the  $N_s$  moves.

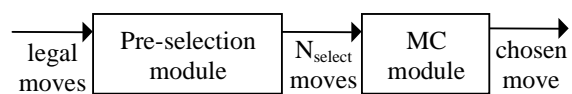


Fig. 1: INDIGO architecture is made up with a pre-selection module and a MC module.

This paper aims to assess two heuristics worth integrating into a MC go program: the History Heuristic (HH) and the Territory Heuristic (TH). They were successfully integrated in the 2004 release of Indigo. They merely consist in using the MC information output from the MC module as input of the two modules as shown in Figure 2. Section 2 describes the current work, i.e. its motivations, and defines these heuristics. Section 3 highlights the experiments. Finally, section 4 provides a conclusion.

## 2. Current Work

First, subsections 2.1 and 2.2 show how to improve the two-step move decision process of Figure 1 by performing a preliminary MC evaluation yielding MC territories. TH consists in using the MC territories. Second, because past MC simulations were already performed on the past positions of the current game, subsections 2.3 and 2.4 show how to use the results of the past MC simulations to improve move selection on the current position, which is HH. Each heuristic can be applied in two ways, external or internal. The term 'internal' means it is used within the random game move selection, and 'external' means it is used within the current game move pre-selection. Thus, we get four heuristics: the External Territory Heuristic (ETH), the Internal Territory Heuristic (ITH), the External History Heuristic (EHH), and the Internal History Heuristic (IHH). Figure 2 shows the aim of the external heuristics: using the available MC information within the move pre-selection module.

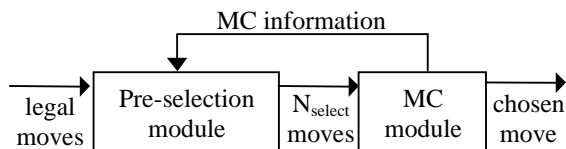


Fig. 2: An external heuristic consists in using the MC feedback within the pre-selection module.

## 2.1. MC evaluation and TH

In addition to the score of a terminal position, the outcome of each intersection (+1 if controlled by Black, -1 if White) is also available. Thus, in addition to the position MC evaluation (that is the mean of terminal position scores [1]), for each intersection, the mean of the intersection outcome is computable. After a few hundred random games, this information can be available with reasonable precision. Hence, it can be used to enhance move selection either within pre-selection (ETH) or within the random games (ITH). A transformation function  $F$  from  $[-1,+1]$  to  $[0,1]$  yields the probability of playing on an intersection, given its expected outcome.  $F(x)=\exp(-kx^2)$  answers the requirement. We set  $P_i=\exp(-k_i)$  for ITH and  $P_e=\exp(-k_e)$  for ETH. Then, the greater the probability of being controlled, the smaller the urgency to play on it.

## 2.2. MC evaluation and ITH

The previous description can be considered as the first iteration of an iterative process. The first iteration consists in performing random games to estimate the intersection expected outcomes. The next iterations launch random games in which the urgency of the moves are multiplied by the probability of playing on its corresponding intersection. After a few iterations, one could expect the MC evaluation and the intersection outcomes to be better estimated than with the one-iteration method. Next, we call TI the number of territory iterations performed by using ITH.

## 2.3. IHH

IHH consists in updating a history number for each move and in using this number during a random game to select a move. It is analogous to the history heuristic defined by [13] in the game tree framework. Within the MC framework, the urgency of a move played during a random game is multiplied by  $\exp(Kv)$ , in which  $v$  is the current evaluation of the move, and  $K$  a constant, inverse of the temperature ( $K=0$  means

$T=\infty$ ). This heuristic was assessed by [7] on 9x9 boards within a pure MC background (very little knowledge). It was also used in [8, 11] with a varying temperature. [7] resulted in a +8 improvement on 9x9 boards. In the current work, the architecture is designed for playing on 19x19 boards, and, consequently, contains a pre-processor that does not allow to update  $v$  for all the legal moves, but only for the  $N_s$  pre-selected moves. Thus, IHH lies out of the scope of the current architecture. However, EHH remains possible.

## 2.4. EHH

A pre-selected move badly ranked by MC on a given position occurring in a game, was repeatedly re-selected and badly re-ranked on the following positions of the game. Therefore, in the flow of a game, the set of pre-selected moves could be filled with moves badly assessed by the MC module. Worse than being pre-selected, these bad moves stayed for a long time in the set of pre-selected moves. As such, the program was not able to get rid of the bad moves from the pre-selection set. The program ran as if  $N_s$  was lowered down to 2 or even 1, which has dramatic consequences.

Hence, it was mandatory to give a penalty to a move badly ranked by the MC module in its previous history. Each move has a history divider whose default value is set up to 1. During pre-selection, the move urgency is set up with the knowledge-based urgency divided by the history divider. For each pre-selected move, the rank determined by the MC module after processing is kept. The history divider of the move ranked by MC on  $N_s-1$  position is multiplied by a number, called *base*. The history divider of the move ranked by MC on  $N_s$  position is multiplied twice by *base*. This way, a bad move is less likely to stay in the pre-selection set. In addition, when a move is actually played in a game, the history dividers of the moves situated at a Manhattan distance inferior to a radius  $R$  from the actual move, are reset to 1. Any move can thus be reset by the last move, and possibly pre-selected again.

## 3. Experiments

For each heuristic, we set up experiments to assess their effect on the level of our programs. One experiment consists in a 400-game match between the program to be assessed and the experiment reference program, each program playing 200 games with Black. Given that the standard deviation of 19x19 games played by our programs is roughly 50 points, 400 games enable our experiments to lower  $\sigma$  down to 2.5

points and to obtain a 95% confidence interval of which the radius equals  $2\sigma$  (i.e. 5 points). We have used 2.4 GHz computers. Whenever the response time of the assessed program varied with the experimental parameters, we mention it. The name of the prototype program is HENRY which stands for Heuristic Ex(iN)ternal terri(his)toRY.

### 3.1. ETH assessment

In this experiment HENRY uses or not ETH. Table 1 shows the improvement resulting from ETH within self-play or against GNU Go 3.2 [9].

	self-play	GNU Go
9x9	+1	-1
13x13	+15	+3
19x19	+43	+10

Table 1: Improvement brought about by ETH within self-play and against GNU Go.

On 19x19 boards, the improvement observed in self-play cannot be overlooked. It is important from a go perspective. However, its importance is less obvious against GNU Go 3.2, which is differently designed, than within self-play. ETH was integrated in the 2004 release of Indigo that played the 2004 Olympiad and won the bronze medal [10].

### 3.2. ITH assessment

HENRY has two parameters: TI, the number of territory iterations, and Pi the probability of playing on intersections controlled by one player (defined in subsection 2.1). HENRY(TI=1, Pi=1) is the reference release. We aim at finding out the best values of TI and Pi. TI may equal 2, 3, 4 or 5. We performed 3 tests. The results of HENRY(TI=n, Pi=0.1) vs HENRY(TI=1, Pi=0.1) are shown in Table 2. The results of HENRY(TI=n, Pi=0.4) vs HENRY(TI=1, Pi=0.4) are given in Table 3. The results of HENRY(TI=n, Pi=0.4) vs HENRY(TI=n, Pi=0.1) are provided by Table 4.

	2	3	4
9x9	-3	+1	-1.5
13x13	-1	+0.5	-4.5
19x19	-7.5	-1	+5.5

Table 2: HENRY(TI=2,3,4, Pi=0.1) vs HENRY(TI=1, Pi=0.1).

Since most of the results are inferior to 5 points, no conclusion can be drawn from Table 2 with sufficient statistical confidence.

	2	3	4
9x9	+3	+2.5	0
13x13	+2	+2.5	+1
19x19	+3	+6	+7

Table 3: HENRY(TI=2,3,4, Pi=0.4) vs HENRY(TI=1, Pi=0.4).

The results given in Table 3 are slightly positive but no certain statistical conclusion can be drawn for most cells. However, by improving the reference program by 6 points with a 10% time overhead, HENRY(TI=3, Pi=0.4) can offer a good compromise.

	2	3
9x9	-0.5	-0.5
13x13	-2.5	-3
19x19	-3.5	+1.5

Table 4: HENRY(TI=2,3, Pi=0.4) vs HENRY(TI=2,3, Pi=0.1).

Since all the results are inferior to 5 points, Table 4 shows that HENRY(TI=2, 3, Pi=0.4) and HENRY(TI=2, 3, Pi=0.1) have approximately the same strength. Summing up the results from Tables 2, 3 and 4 to find out a statistical conclusion remains difficult. Nevertheless, taking the small time overhead (10%) due to processing territory iterations, and using ITH with (TI=3, Pi=0.4) can be worth considering on 19x19 boards (about 6-point improvement) but not really on 9x9 or 13x13 boards.

### 3.3. EHH assessment

In this subsection, HENRY has two parameters: *base* and *R*. HENRY(base=1, R=19) is the reference release. In a first stage, we aim at finding out the best values of *base* and *R*, with *base*=2, 5 or 10 and  $5 \leq R \leq 13$ .

	5	7	9	11	13
2	+9	+8	+7	+4	0
5	0	+4	-1	-3	+3
10	-6	-5	-4	-10	-8

Table 5: HENRY(base=2,5,10,  $5 \leq R \leq 13$ ) vs HENRY(base=1, R=19).

Because this was a first stage experiment, we did 100 games per confrontation only to obtain approximate results. The experiment was performed on 19x19 boards. Table 5 yields the results showing that  $base=2$  remains the best choice. Then, in a second stage, given that  $base=2$ , we aimed at determining the best value for  $R$ , performing 400 games per confrontation again. Table 6 yields the results.

	2	3	4	5	6	7	8	9
mean	+5	+9	0	+7	+9	+12	+9	+5

Table 6: HENRY( $base=2$ ,  $2 \leq R=11$ ) vs HENRY( $base=1$ ,  $R=19$ ).

Due to a different number of games per confrontation, the results given in Table 6 are more accurate than the results given in Table 5. The result obtained by HENRY( $base=2$ ,  $R=4$ ) is amazing because it shows a discontinuity that we are not able to explain. HENRY( $base=2$ ,  $R=7$ ) gives the best improvement (+12 points), which is significant, but not extraordinary from go standards. Against GNU Go 3.2, the improvement amounts to +5 points, smaller than in self-play.

## 4. Conclusion

In this paper, we have defined the four following heuristics, IHH, EHH, ITH and ETH within the MC go framework, and we have experimentally evaluated the playing level improvement brought about by their integration within INDIGO, either in self-play or against GNU Go 3.2.

ETH is the most effective heuristic, enabling Indigo to improve by 40 points on 19x19 boards in self-play without time overhead. The improvement brought about by ETH is smaller against GNU Go 3.2 (+10 points) but still significant. ITH is interesting but not effective enough (+5 points) to compensate the time overhead (+10%). IHH cannot be taken into account in our pre-selection framework. Finally, EHH brought about a 12-point improvement within self-play and a 5-point improvement against GNU Go 3.2. The external heuristics use the information processed by the MC module as a feedback within the pre-selection module as shown by Figure 2. These heuristics enabled Indigo to perform well (bronze medal) at the 2004 Computer Olympiad on 19x19 boards [10].

Regarding the scope of Monte Carlo go, we have several interesting perspectives: re-produce the hints of [15], use reinforcement learning [16] to improve the quality of move urgencies in random games, and study local MC search.

## 5. References

- [1] B. Abramson. Expected-outcome: a general model of static evaluation. *IEEE Transactions on PAMI*, 12:182-193, 1990.
- [2] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of Poker. *Artificial Intelligence*, 134:201-240, 2002.
- [3] B. Bouzy, The move decision process of Indigo, *ICGA Journal*, 26(1):14-27, March 2003.
- [4] [www.math-info.univ-paris5.fr/~bouzy/INDIGO.html](http://www.math-info.univ-paris5.fr/~bouzy/INDIGO.html)
- [5] B. Bouzy, Associating knowledge and Monte Carlo approaches within a Go program. *Information Sciences*, 2005, 11 pages, article in press.
- [6] B. Bouzy and T. Cazenave, Computer go: an AI oriented survey, . *Artificial Intelligence*, 132:39-103, 2001.
- [7] B. Bouzy and B. Helmstetter, Monte Carlo developments, In E. Heinz J. van den Herik H Iida, editors, 10<sup>th</sup> Advances in Computer Games, pages 159-174, Graz, 2003. Kluwer Academic Publishers.
- [8] B. Brüggmann, Monte Carlo go, [www.joy.ne.jp/welcome/igs/Go/computer/mcgo.tex.Z](http://www.joy.ne.jp/welcome/igs/Go/computer/mcgo.tex.Z), 1993.
- [9] [www.gnu.org/software/gnugo/devel.html](http://www.gnu.org/software/gnugo/devel.html)
- [10] D. Fotland, Go Intellect wins 19x19 go, *ICGA Journal*, 27(3):169-170, September 2004.
- [11] [www.ideanest.com/vegosl](http://www.ideanest.com/vegosl), 2003.
- [12] T. Raiko, The go playing program called Go81, FAIC 2004, pages 197-206, Helsinki, Finland, September 2004.
- [13] J. Schaeffer, The history heuristic and alpha-beta search enhancements in practice. *IEEE Transactions on PAMI*, 11(11):1203-1212, November 1989.
- [14] B. Sheppard, World-championship-caliber Scrabble, *Artificial Intelligence*, 134:241-275, 2002.
- [15] B. Sheppard, Efficient control for selective simulations, *ICGA Journal*, 27(2):67-80, June 2004.
- [16] R. Sutton and A. Barto, *Reinforcement Learning: an introduction*. MIT Press, 1998.
- [17] G. Tesauro and G. Galperin, On-line policy improvement using Monte Carlo search. In *Advances in Neural Information Processing Systems*, pages 1068-1074. Cambridge, MA, 1996, MIT Press.