Technical Report

On the Laziness of Monte-Carlo Game Tree Search in Non-tight Situations

September 8, 2008

Ingo Althofer
Institute of Applied Mathematics
Faculty of Mathematics and Computer Science
Friedrich-Schiller University Jena
07740 Jena - Germany

Email: althofer@minet.uni-jena.de

Key words: Monte-Carlo game tree search, computer Go, experimental analysis.

1 Introduction

In many different games, including the heavy-weight classic Go, Monte-Carlo game tree search performs surprisingly well ([Bru 1993], [Cou 2006], [Gel 2007]). Already in several situations it was observed that Monte-Carlo (and variants and extensions of it) seems to play really well when the situation in the game is tight. In non-tight situations, however, Monte-Carlo (called MC for short in the sequel) often shows some laziness.

In this report we investigate the very basic MC algorithm in a class of very simple abstract games. Our experiments indicate that MC is at its best when the board situation is balanced or even slightly worse for the player to move.

In Section 2 we introduce "Double Step Races", a class of very simple board games without chance and without interaction. The very short Section 3 gives the straightforward optimal strategy for Double Step Races. The basic MC algorithm is described in Section 4. Section 5 specifies the settings for our experiment, Section 6 contains the results and Section 7 the evaluation. We conclude with a short discussion in Section 8.

2 Rules of the Game "Double Step Race"

There are two players, Black and White. The players move in turn, Black has the first move. Each player has only one stone. This stone is running in a horizontal lane, consisting of finitely many squares. The stone is starting in the leftmost square of its lane. A legal move consists of moving the stone one or two steps to the right in the lane. The stones of the players are running in different lanes. They do not interfere. The player who first reaches the rightmost square in his lane is winner. Naming: when the lane of player Black contains k+1 squares (incldùing starting square and goal), and the lane of player White has m+1 squares, we call the game DSR--k-vs-m, which is short for "Double Step Race with two players and lanes of lengths k and m".

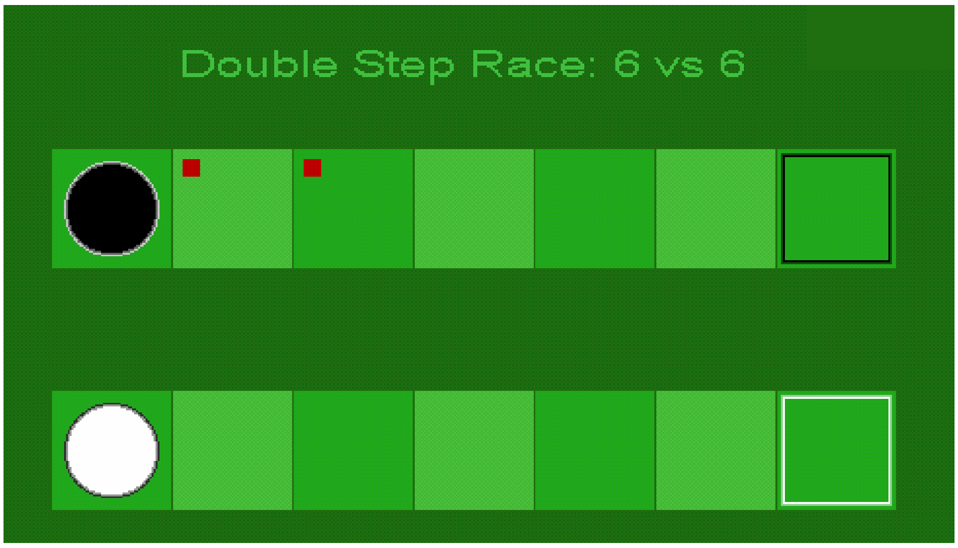

Diagram 2.1 The board for DSR--6-vs-6, including the feasible target squares for the first move of Black.

3 Perfect Play in Double Step Races

In each situation it is optimal to move the stone two squares to the right. (The only exception is when the distance between stone and goal is an odd number. Then it is equi-optimal to make a single step in the first move. Of course, in the square next to the goal the single step move is the only feasible move.) Looking at DSR--6-vs-x, Black has a forced win for all x larger or equal to 5. White has a win for all x smaller than 5.

4 Specification of Monte Carlo Game Tree Search

In our experiment we will use the most basic version of MC game tree search: In a given situation S all feasible moves are determined. For each move n random games are played. The move with the best score (from the viewpoint of the player who has to move in S) is executed. "n" is called the Monte-Carlo parameter of this algorithm MC(n).

5 Setup of the Experiment

In DSR--6-vs-x only for the "intermediate" values from 3 to 10 for x the outcome of the game depends on the skills of the players. We looked exactly at these games. For each such "6-vs-x" rule and each MC parameter n from {1, 2, 4, 8, 16, 32, 64} 10,000 games were played where both players used MC(n) to decide upon their moves.

6 Results of the Experiment

Table 1: Number of wins for Black (from 10,000 games)

n	6-vs-3	6-vs-4	6-vs-5	6-vs-6	6-vs-7	6-vs-8	6-vs-9	6-vs-10
1	0361	2204	4805	6933	9340	9649	9967	9988
2	0193	1987	5799	6970	9654	9752	9991	9996
4	0047	1468	7039	7450	**9905**	**9868**	9999	9999
8	0002	0836	**8573**	**8228**	**9989**	**9969**	**10000**	**9999**
16	0000	0286	**9557**	**9264**	**10000**	**9992**	10000	10000
32	0000	0040	**9936**	**9875**	10000	10000	10000	10000
64	0000	0000	**10000**	**9991**	10000	10000	10000	10000

Table 2: Number of games with a (bad) single step in the first move of Black

n	6-vs-3	6-vs-4	6-vs-5	6-vs-6	6-vs-7	6-vs-8	6-vs-9	6-vs-10
1	4714	3992	**3511**	**3732**	4259	4798	4935	4954
2	4363	3270	**2797**	**2993**	3625	4493	4803	4955
4	3864	2368	**2091**	**2081**	2870	3988	4755	4897
8	2968	1319	**1131**	**1281**	1835	3331	4502	4920
16	1775	0568	**0396**	**0465**	0831	2399	4185	4912
32	0642	0118	**0064**	**0086**	0250	1334	3395	4687
64	0080	0009	**0000**	**0006**	0032	0438	2339	4453

7 Evaluation of the Experiment

In each column of Table 1 the scores change monotonically. For x = 3 and 4 the scores decrease towards the game-theoretic value 0. For all x > 5 the scores increase towards the game theoretic value 10,000.

In each row the scores are monotonically increasing, at least in most cases. However, when comparing the columns 6-vs-5 and 6-vs-6, Black turns out to perform better in the more difficult situation (6-vs-5) for MC parameters 8 and larger. This may come as a surprise: Both games are won for Black in case of perfect play, however in 6-vs-5 the opponent is already nearer to his goal, so the situation is more tight for Black. The pairs of games (6-vs-7, 6-vs-8) and (6-vs-9, 6-vs-10) show similar performances (relevant parts printed in fat). This phenomenon is related to the phenomenon observed in Table 2.

On Table 2: Single steps in the very first moves are sometimes deadly bad moves: for x = 5 and 6 they turn a won position into a lost position. In the other situations they are only "normally bad", because they decrease Black's chances in a game between non-perfect players.

In each column the values are decreasing monotonically, when the MC parameter n becomes larger. This is not at all a surprise: the larger n, the better the MC algorithm.

In each row of the table the values (standing for the frequency of bad or lazy first move) are first decreasing and then increasing again. In each row the minimum is achieved either for x=5 or x=6.

8 Discussion

* We did not investigate "Double Step Races" (DSR) because they are interesting games. Just to the contrary, we invented the trivial DSR to have games as simple as possible for investigating the **laziness of Monte-Carlo** game tree search in non-tight situations.

* The laziness of MC for non-tight instances of Double Step Races is rather easy to explain: in these instances **all** feasible moves get either scores near to 1 (in positions with clear advantage) or near to 0 (in positions where the player is clearly behind). This makes it more likely that a non-optimal move gets the best MC score and is thus selected instead of the best move.

* Procedures like "dynamic komi" (for instance in computer Go) may try to give MC "always" the impression that the situation is tight, resulting in more goal-oriented play.

* In a forthcoming paper, we will report on more general experiments with Monte-Carlo for Double Step Races: other lane lengths, MC with heuristic games instead of completely random games, non-independent MC runs, more than two players, asymmetric move rules, ...

Acknowledgements

Sincere thanks go to Joerg Sameith. When duddling around with his nice multi-game program "McRandom" I stumbled upon the phenomenon of MC-laziness. Thanks to Bernd Bruegmann, just for being there at Jena University. He is no longer active in computer go research, but has a chair in theoretical physics. Always, when coming along on the floor at his office I get fresh motivation to think about Monte Carlo game tree search. Thanks also to Gian-Carlo Pascutto for encouraging me to write this quick report by his reply in the computer-go mailing list. (Quote: "However, there is less and less doubt in my mind about that the effect you describe exists and it hurts playing strength", September 8, 2008).

References

[Bru 1993] B. Bruegmann. Monte Carlo Go. Technical report, 1993, downloadable from http://www.ideanest.com/vegos/MonteCarloGo.pdf .

[Cou 2006] R. Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In "Computer Games 2006" (Editors P. Ciancarini and H. J. van den Herik). Paper downloadable from http://remi.coulom.free.fr/CG2006/CG2006.pdf

[Gel 2007] S. Gelly. A contribution to reinforcement learning; application to Computer-Go. Ph.D. thesis, 2007,
downloadable from http://www.lri.fr/~gelly/paper/SylvainGellyThesis.pdf

[Sam 2005] J. Sameith. Website http://www.joerg.sameith.net/denken.html on his Monte Carlo game programs Hanfried, McRandom, and Turbo RapaNui; since 2005.