

# Ph D in Computer Games and Artificial Intelligence

Bruno Bouzy  
C.R.I.P.5

UFR de mathématiques et d'informatique  
Université René Descartes, Paris  
bouzy@math-info.univ-paris5.fr

May 10, 2006

## 1 Field of Study

The field of study of the PhD position offered is Computer Games (CG) and Artificial Intelligence (AI) in general. More precisely the games studied are zero-sum two-player complete information games with a high complexity such as Go, Amazons or Shogi. In addition to Tree Search (TS) and knowledge-based techniques, the technique newly used in the computer Go community is Monte-Carlo (MC) [5]. The first aim of the PhD thesis is to explore Machine Learning techniques (ML) such as Evolutionary Computation (EC) or Reinforcement Learning (RL) using high complexity games as testbeds. The second aim is to build strong game playing programs in using ML techniques.

To illustrate what can be expected during this PhD thesis, this section explains the field of study in a Go oriented way: Computer Games, Computer Go, Monte-Carlo Go, RL for Computer Go, and Evolutionary Computation for Computer Go. However, the game of Amazons, not described here, is also under investigation in our laboratory.

### 1.1 Computer Games

Computer Games has witnessed the enhancements done in AI for the past decade [16], and future improvements will surely go on in the next decade [22]. For instance, in 1994, Chinook beat the Checkers world champion [14]. In 1997, Deep Blue [7] beat Garry Kasparov, the Chess world champion. In 2006, solving Checkers is nearly achieved [15]. In Chess, the best programs rank on a par with the best human players. In Shogi, the best programs rank the grand-master players [10]. Moreover, combinatorial complexity of a game can be estimated with the game tree size that, in turn, can be estimated by  $B^L$ , where  $B$  is the average branching factor of the game, and  $L$  is the average game length. Table 1 provides the values of  $B^L$  for these games, and Go.

By observing that the best Go programs are medium on the human scale, at least far below the level of the best human players, a correlation between the size of the

Game	Checkers	Chess	Shogi	Go
$B^L$	$10^{32}$	$10^{123}$	$10^{226}$	$10^{700}$

Table 1:  $B^L$  Estimation.

game tree and the playing level of the best programs human on the human scale can be noticed. The game tree search paradigm accounts for this correlation. A classical game-tree-based playing program uses a tree-search and an evaluation function. On current computers, this approach works well for Checkers, Chess, and Shogi. In these games, the search is sufficiently deep, and the evaluation function sufficiently adequate to yield a good result. On the contrary, the Go tree is too huge to allow a good result. Furthermore, the evaluation function on non terminal positions is not well known, and position evaluations are often very slow to compute on nowadays computers.

## 1.2 Computer Go

Since 1990, an important effort has been done in computer Go. The main obstacle is to find out a good evaluation function [13]. Given the distributed nature of this game, it was natural to study the breakdown of a position into sub-parts, and to perform local tree searches using intensive pattern-matching and knowledge bases [12, 4]. The best programs are commercial : Many Faces of Go, Handtalk, Go++, and Haruka, a Japanese software. In 2002, GNU Go, an open source program, became almost as strong as these programs. Various academic programs exists as well.

## 1.3 Monte-Carlo Go

Monte-Carlo is appropriate to games containing randomness, for example to Backgammon in which the players throw dice. In Backgammon, simulations are used after the games or at learning time [21] to find out new policies. Monte-Carlo is also adapted to games including hidden information such as Poker [2] or Scrabble [17]. For complete information games, simulations can be appropriate as well. Abramson proposed a Monte-Carlo model for such games [1]. To choose a move on a given position, the corresponding idea is the greedy algorithm a depth one. For each move on the given position, launch a given number  $N$  of random games starting on this position, score the terminal positions, and average all the scores, and finally play the move with the best mean. The obvious upside of MC is its low complexity when  $B$  and  $L$  are high:  $O(B^L)$  for tree search, and  $O(NBL)$  for Monte-Carlo. In the early nineties, Bernd Brügmann succeeded to develop the first 9x9 MC Go program, Gobble [6]. The MC approach has gained popularity in the computer Go community since 2002. The reasons lie in the speed of current computers, and in the easy development of MC Go programs.

## 1.4 Reinforcement Learning and Monte-Carlo Go

Instead of using the uniform probability in random games, this approach consists in using a non uniform probability resulting from the use of domain-dependent knowl-

edge. Experiments made on 9x9, 13x13 and 19x19 boards show impressive results of the knowledge approach associated to the MC approach [3]. The results clearly show that using a domain-dependent probability is superior to using a uniform probability. It is now natural to look toward automatic methods to build the knowledge. This leads the PhD to possibly use RL in a MC Go architecture.

The goal would be to automatically build a random player used in the Monte-Carlo architecture with RL. RL is deeply marked by Richard Sutton's work. Sutton is the author of Temporal Difference (TD) method [19], and co-author with Barto of a book describing the state of the art [20] (also described by [11]). RL is known by the success of Q-learning [23]. RL formalism is the one of Markov Decision Process (MDP): an agent evolves in a non-deterministic environment. He performs actions according to its own policy. His actions make him changing from state to state, and result in returns. The aim of the agent is to maximize his cumulated return over the long term.

### **1.5 Evolutionary Computation and Computer Go**

Evolution Computation (EC) principles [9] have been used for computer Checkers [8] to build a Checkers program, Anaconda, that achieved good results on the human scale. These results can be used in computer Go as well. Several works already exists, and they consists in making a neural net evolve [18].

## **2 Program of Study**

The program of study described in this section is preliminary. To sum up the previous section, the research addresses the problem of how machine-learning methods can be used to build good evaluation functions for computer games, i. e., computer Go, computer Amazons or computer Games in general. The research cannot ommit the main computer game paradigm: tree search. The following research questions will be addressed:

- Can we devise intelligent learning methods for computer games in general, and in computer Go or computer Amazons in particular ?
- Can efficient search techniques used for computer Shogi can be applied to computer Go or computer Amazons ?
- How can search-based methods be combined with machine learning techniques for such games ?
- Can the proper implementation of a newly created method improve the level of existing game-playing programs ?

The program below assumes a three-year PhD thesis, which sounds a reasonable possibility.

## 2.1 First Year

Broad investigation of known techniques for ML and computer Go, computer Shogi and computer Amazons. Investigating the features of game problems for which new techniques should be developed. One publication in a conference related to computer games should be a final goal of this first year.

## 2.2 Second Year

Developing specialized intelligent methods for the game problem chosen. Building and optimising a game program for testing the game problems. One conference and one journal publication can be expected.

## 2.3 Third Year

Refining the program. Attending international computer game competition. Writing the PhD manuscript. One conference and one journal publication can be expected.

## 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. Associating knowledge and Monte Carlo approaches within a go program. *Information Sciences*, 175(4):247–257, November 2005.
- [4] B. Bouzy and T. Cazenave. Computer go: an AI oriented survey. *Artificial Intelligence*, 132:39–103, 2001.
- [5] B. Bouzy and B. Helmstetter. Monte Carlo go developments. In Ernst A. Heinz H. Jaap van den Herik, Hiroyuki Iida, editor, *10th Advances in Computer Games*, pages 159–174, Graz, 2003. Kluwer Academic Publishers.
- [6] 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.
- [7] M. Campbell, A.J. Hoane, and F-H Hsu. Deep blue. *Artificial Intelligence*, 134:57–83, 2002.
- [8] K. Chellapilla and D. Fogel. Evolving neural networks to play checkers without relying on expert knowledge. *IEEE Trans. Neural Networks*, 10(6):1382–1391, 1999.
- [9] D. Goldberg. *Genetic Algorithms in Search, Optimization, and Machine Learning*. Addison-Wesley Publishing Co, 1989.

- [10] H. Iida, M. Sakuta, and J. Rollason. Computer shogi. *Artificial Intelligence*, 134:121–144, 2002.
- [11] L. P. Kaelbling, M. Littman, and A. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.
- [12] M. Müller. Computer go. *Artificial Intelligence*, 134:145–179, 2002.
- [13] M. Müller. Position evaluation in computer go. *ICGA Journal*, 25(4):219–228, December 2002.
- [14] J. Schaeffer. *One Jump Ahead: Challenging Human Supremacy in Checkers*. Springer-Verlag, 1997.
- [15] J. Schaeffer, Y. Björnsson, N. Burch, A. Kishimoto, M. Müller, R. Lake, P. Lu, and S. Sutphen. Solving checkers. In *IJCAI*, pages 292–297, 2005.
- [16] J. Schaeffer and J. van den Herik. Games, Computers, and Artificial Intelligence. *Artificial Intelligence*, 134:1–7, 2002.
- [17] B. Sheppard. World-championship-caliber scrabble. *Artificial Intelligence*, 134:241–275, 2002.
- [18] K. Stanley and R. Miikkulainen. Evolving a roving eye for go. In *Genetic and Evolutionary Computation Conference*, New-York, 2004.
- [19] R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.
- [20] R. Sutton and A. Barto. *Reinforcement Learning: an introduction*. MIT Press, 1998.
- [21] 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.
- [22] H.J. van den Herik, J.W.H.M. Uiterwijk, and J. van Rijswijck. Games solved: Now and in the future. *Artificial Intelligence*, 134:277–311, 2002.
- [23] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.