

Toward a go player computational model

Accreditation to supervise research document

Bruno Bouzy

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 document describes computer go researches carried out since 1997 at the university of Paris 5 to obtain the accreditation to supervise research. Computer games have witnessed improvements performed in AI. However, go programming constitutes a very hard task for two reasons. First, the size of the game tree forbids any global tree search, and second, a good evaluation function is very hard to find out. The research presented within the Indigo project aims at producing a go program "as strong as possible" while publishing the results. Between 1997 and 2002, the approach was based on knowledge, and had many links with many AI sub-domains. The scientific results are described for each sub-domain: DAI, cognitive science, fuzzy logic, uncertainty representation, combinatorial games, retrograde analysis, tree search, evaluation function and spatial reasoning. In 2002, the Monte Carlo approach was launched, and gave promising results. This fact brought about the programming of an approach associating Monte Carlo and knowledge in 2003. Then, that association yielded good results at the computer olympiads held in Graz in 2003. Thus, the Monte Carlo approach will be pursued in the future, and will be associated with bayesian and reinforcement learning.

1 Introduction

As an associate professor in computer science at Paris 5 university, I have worked on computer go for thirteen years. In fact, after working in a data processing company from 1986 to 1990 as a computer engineer, I was very fond of the game of go, and I started a PhD about computer go in 1991. Its title was “Cognitive modeling of the go player”. I defended my thesis in 1995 under the supervision of Jacques Pitrat. At that time, an early release of Indigo was developed. Since 1997, I have worked in the SBC (Systèmes à Base de Connaissances) group of the IAA (Intelligence Artificielle et Applications) team, directed by Dominique Pastre, within the CRIP5 (Centre de Recherche en Informatique de Paris 5), the laboratory directed by Jean-Marc Labat and associated to the computer science and mathematical department led by Dominique Seret. I have carried out computer go researches, which released a second version of Indigo, much stronger than the first one. Over these years, the results have been published in national and international conferences and journals. The aim of this document is to describe these researches in order to obtain the accreditation to supervise research projects at university level.

To this end, different viewpoints are worth considering. Section 2 presents the context, the aim and the method of this work, and explains why this document is entitled “Toward a go player computational model”. Section 3 provides a chronological overview. Section 4 classifies the publications and comments upon the main ones. Section 5 presents our work for each AI sub-domain. For each sub-domain, the background is shown and the results of our work are highlighted. Section 6 provides a computer game description to show our contribution to this domain. Section 7 describes the other activities linked to academic research. Before conclusion, section 8 describes the future perspectives of my work.

Most of the publications referred to in this document can be directly accessed on the web with pre-prints at <http://www.math-info.univ-paris5.fr/~bouzy/publications.html>. This document is the English version of the reference document written in French language.

2 Context, method and aim

This section describes the background of our research. First, it highlights the interest of producing go programs for AI. Second, it presents the method to assess the program’s improvements. Then, it shows how this research can be valuable within the academic context. Moreover, by mentioning the difference between a computational model and a cognitive one, it explains why this document is entitled “Toward a go player computational model”. Finally, this section focusses on our goal and on the method to reach it.

2.1 Why a go program ?

Mind games are appropriate application fields for AI and computer science. They witness the improvements made in AI [129]. Tree search is the most important paradigm in computer games. The smaller the game tree is, the more relevant the tree search techniques are, and the better the game playing software is [53]. In othello, with its limited complexity (10^{58}), the best program playing level lies beyond human abilities. In chess which is more complex (10^{123}) than othello, the best programs obtain a level similar to the best human players. In go which is very complex (10^{700})¹, the best programs have achieved an average level only on the human scale. Unlike in chess and othello, global tree search does not work well in go. The game of go is an obstacle for computer science, which brings about the necessity for AI to improve its methods [55]. Our aim is to develop a go playing program *as strong as possible* whose name is Indigo. The 2002 release can be downloaded on the web [44]. Two papers introduce its global architecture [30, 49].

2.2 How to assess the improvements ?

In order to obtain a go program *as strong as possible*, we decided to adopt a yearly approach. Each year, a new release is produced. For instance, Indigo99 was released in 1999. To improve the program, a working release is created. As long as our work progresses during the current year, it is compared to the previous year's release. A comparison consists in several games played out on boards of different sizes. It can include handicap games as well. The result of a comparison is either a handicap stone number or an average score. The standard deviation of the score of a set of games is about 50 on 19x19. Thus, one hundred games are necessary to get a five-point precision on the result with about 65% of confidence only. In other words, a comparison is time consuming. But such an exhaustive comparison is necessary to be certain of our breakthroughs. Thus, each year, the "internal" improvements are assessed. For instance, Indigo2000 is four stone stronger than Indigo99. However, the real test is carried out during computer competitions when Indigo plays against other programs. For instance, 2003 was a very good year. First, our internal tests showed that Indigo2003 was 40 points better than Indigo2002. Second, at the last computer olympiads held in Graz, Indigo 2003 ranked 4th out of 10 participants in the 9x9 competition and 5th out of 11 in the 19x19 competition [5], which publicly confirmed the improvements made.

2.3 Publishing the research results

This subsection shows how our current work can be evaluated not only within the computer go community but also within the academic world. To this end, it is important to start by mentioning the three categories of go programmers. The first category gathers the authors of the best programs in the world. Their

¹approximation of 361!

programs are commercial and they earn important benefits from their work. Unfortunately for the scientific community, the designs of these programs are kept secret. The second category corresponds to the free software world. The sources of such programs are public and anyone may contribute to the general effort. The third category belongs to the academic world in which the researchers develop programs described in scientific publications. I belong to this category, and part of my *work consists in publishing* on a regular basis, which is not an easy task. In the beginning, my strategy was to publish in national conferences and now to move towards more prestigious conferences or journals as long as the work progresses (cf <http://www.math-info.univ-paris5.fr/~bouzy/publications.html>). Along with the publication effort, another very important point has to be highlighted. The success of Deep Blue over Kasparov in 1997, was the strongest mark in the computer game history. Apart from its impact on the society at large, its main effect within the computer game community was to shed light on other games, particularly on more complex games than chess, such as go, shogi and Chinese chess. This brought about a series of important academic changes. First, the ICCA (International Computer Chess Association) transformed itself into ICGA (International Computer Game Association) [2]. Moreover, new conferences such as “Computers and Games” have been set up since 1998. Besides, the conference “Advances in Computer Chess” was extended to “Advances in Computer Games”. Furthermore, since 2000, the computer olympiads have been held each year. To sum up, the publications obtained over these last years illustrate two facts: on the one hand, the positive results of our work, and on the other hand, the new interest of the academic community for other games than chess.

2.4 Cognitive or computational model ?

The title of my PhD was “Cognitive modeling of the go player”. The work currently performed is different but, to some extent, is a continuation of the PhD work. Thus, nine years later, it is interesting to find out a relevant title to this document, expressing both this continuity and these differences. To this aim, it is important to know whether the Indigo’s conceptual model was either computational, or cognitive, or both, at the time of the PhD and now.

At that time, while the model was implemented on the computer, it was obviously computational. However, because the background of my work was cognitive science, my aim was also to provide a cognitive model. Thus, the thesis highlighted a correspondence between the concepts situated in the software (the computational concepts) and the concepts used by human verbalizations (the cognitive concepts). An important goal was to keep a strong connection between our work and the human strategy. In this view, the model was cognitive. Thus, in 1995, we may say that the model was both cognitive and computational.

Since then, the work has been performed in a computational perspective: an idea being given, it is necessary to specify it, foresee its results, program it, perform the tests, gather the results, compare them to the expected ones and conclude on the relevance of the idea and its future exploitation. Although

the author of the program may be an expert in the domain, and the initiator of the ideas, the method consists in letting the computer experiment results decide whether the idea is good or not. In this view, the program obtained in 2004 hardly corresponds to a cognitive model, but it is strongly linked to a computational model. Besides, Indigo has a large margin for improvements, and the corresponding model is not finished yet. Such are the reasons for entitling this document “Toward a go player computational model”.

2.5 Aim

To sum up, the goal of this work has been to release a go program *as strong as possible*, Indigo, corresponding to a *computational model*. To evaluate this work, we assess the yearly improvements with the *computer go competition results* and we provide the computer game and AI communities with *publications* in conferences and journals.

2.6 Method

Our research method lies on two principles. The first one is practical and is described in subsection 2.2. Its summary consists in permanently having a prototype attending the competitions. The second one is didactical. So as to find something interesting, a researcher has to stand on the boundaries of his domain. To reach them, he has to know the interior first. Thus, he must be careful to perform experiments not only to find out something new, but also to learn existing techniques in his own domain. For instance, in 2001, after concentrating on knowledge approaches on 19x19 boards, it was interesting to lead our work towards small boards, and adopt classical tree search techniques. This was done not only to see the results of tree search approaches on small boards, but also to learn about them, particularly about alfa-beta enhancements. The same idea was underlying the experiments performed in sub-domains such as retrograde analysis or Monte Carlo. Whenever a technique is used, the goal is twofold: knowing whether the technique is adapted to the problem, and at least mastering the technique in practice.

3 Chronological overview

This section gives a chronological overview of our work and shows how our research method described in subsection 2.6 led us to reach our goal defined in subsection 2.5. Since 1991, the various periods have been: the PhD thesis period (1991-1995), the post-doctoral period (1995-1997), two research periods (1997-1999 and 2000-2002) as an associate professor at Paris 5 university, in the SBC group headed by Dominique Pastre, and finally the year 2003.

3.1 1991-1995

Over those years, interested by computer go and AI [55], I carried out a PhD about cognitive modeling of the go player [32], directed by Jacques Pitrat in his team Métaconnaissances [115], within the LAFORIA laboratory at the university of Paris 6. The Indigo's implementation led to the validation of the cognitive model. The first release of Indigo was written in C++ between 1991 and 1993. Four abstraction levels were designed: the global level, the “group” level, the shape level and the tactical level [30]. At that time, Indigo could be compared to an expert system without any global search. Indigo performed tactical tree searches, used shapes, mathematical morphology and life and death knowledge. Finally, it assigned a priority to each point on the board, and it played the move with the highest priority. Indigo did not verify whether the chosen move was effective or not.

3.2 1995-1997

During the post-doctoral period, the A.T.E.R. position obtained at Paris 6 university gave me the opportunity to write papers describing our thesis's work in various domains of AI: Distributed AI (DAI) [37, 39], cognitive science [34], fuzzy sets [31], uncertainty representation [36], incremental reasoning [38], and contextual reasoning [52].

3.3 1997-1999

In 1997, the position obtained as an associate professor at Paris 5 university enabled me to re-write Indigo completely. The four level description of 1995 was simplified, and became the Indigo evaluation function. The process of choosing the move was split into two parts. If the position was calm, Indigo used depth-one global tree search. Otherwise, it took the best urgent move, the urgency being defined with complex domain-dependent knowledge. For the first time, Indigo attended an international competition, the 1998 Ing Cup, and was ranked 10th out of 17 programs, which was a satisfactory result for a first attempt. In 1999, the functions were optimized in terms of time so as to perform global quiescence tree search [16] instead of depth-one tree search. As a result, Indigo99 won 80% of the games played against Indigo98, and ranked 13th out of 16 at the 1999 Ing Cup. A paper about the complexity of sub-games of go was written [40] at that time.

3.4 2000-2002

The year 2000 In 2000, GNU Go [58], the go program of the Free Software Foundation became strong enough. To make it play automatically against Indigo by mean of an arbiter program, the source was modified, and Indigo's tests were enriched with a strong new opponent. Thanks to the set of games played against GNU Go-2.4 that was two stones stronger than Indigo, Indigo2000 became four

stones stronger than Indigo99. Then, Indigo attended the 2000 Mind Sport Olympiad in London, and was ranked 5th out of 6 programs. Nick Wedd, organizer and referee of this competition, describes a feature of the current state of computer go [159]. At the end of one of the two games played between Aya [164] and Indigo, both programs passed. However, according to the human go players watching the game, one black group was alive if Indigo played first, and dead if Aya played first. The two programs could not see this, and passed, while the issue of the game depended on it. For the referee, the problem was to decide who the winner was. Between human players, this problem is generally solved through discussion and playing until an agreement is reached. Between artificial players, no protocol exists and the referee must decide. In fact, Indigo thought the black group was dead, and I honestly resigned this game, which lightened the referee from a burden.

The year 2001 In 2001, the didactical method described in subsection 2.6 was applied. Experiments about tree search [41] and retrograde analysis [42] were carried out on very small boards. Meanwhile, researches about spatial reasoning in go were pursued and completed [43]. Furthermore, the important paper describing the AI oriented survey of computer go which was started with Tristan Cazenave two years before, was completed and published in AI Journal [53].

The year 2002 In 2002, the response time of the program was sufficiently optimized to start experiments about generalized life and death problems. Generalized life and death problems deal with incompletely circled groups. Thanks to the newly developed life and death problem solver, Indigo was stronger at solving life and death problems arising in actual games, which was to be expected. Unfortunately, this life and death module did not enable Indigo to play better on whole games, and we did not integrate this life and death tree search module into Indigo2002. This was an example of the counter-intuitive fact that adding a new module to a program does not necessarily make it play better. As a result, Indigo2002 is a fast program playing its whole 19x19 game in two minutes on a 450 MHz computer. It can be downloaded on the web. [49] describes the move decision strategy used by Indigo at that time. In July, this release participated in the second 21st century cup [4] held in Edmonton and was ranked 10th out of 14 participants [47]. In October, this release ranked 6th out of 10 programs in the computer go festival organized in Guyang, China. Meanwhile, we wrote two other papers. The first [45] deals with evaluation functions using either dimensional concepts or metric ones. The second [48] describes the application of mathematical morphology to computer go.

3.5 The year 2003

The year 2003 showed the first results of experiments about Monte Carlo. In 2001, Bernard Helmstetter, PhD student of Tristan Cazenave at Paris 8 university, wrote a go program, Oleg, based on Brügmann's Monte Carlo go [57]. This

convinced me to start a Monte Carlo experiment too. Thus in 2003, Bernard Helmstetter and I had two Monte Carlo go programs, Oleg et Olga, that could play on 9x9 boards in a reasonable time. Although they did not contain domain-dependent knowledge, they ranked on a par with Indigo2002, which was amazing [54]. Thus, I developed a new version of Olga integrating domain-dependent knowledge. The association between Monte Carlo and knowledge takes on two aspects. First, the random games contain small patterns advising moves, giving more significance to the score of terminal positions. Second, the Monte Carlo module works on moves that are output of a pre-processor heavily based on domain-dependent knowledge, speeding up the whole process and enabling the program to play 19x19 games in reasonable time on current computers. This double-sided construction was published in [46]. It gave good results at the computer olympiads in Graz. Indigo was ranked 4th out of 10 programs in the 9x9 competition [153], and 5th out of 11 in the 19x19 competition [73]. This was the first time that Indigo had been ranked in the first half of an international competition. This result should be confirmed in the coming competitions. Our current conclusion is that Monte Carlo is worth continuing.

4 Publications

This section comments upon major publications of our research and classifies all our publications. Most of the publications referred to in this document can be accessed on the web (<http://www.math-info.univ-paris5.fr/bouzy/publications.html>) in a preprint form.

4.1 Main publications

In this section, five publications have been selected on two criteria. First in terms of a faithful illustration of my work. Second in terms of academic recognition. In my case, these two aspects cannot be dissociated. In short, whenever a paper illustrated my work, I carried out the effort from a publication in a national conference to more prestigious international conferences or journals with high academic recognition. For instance, the Monte Carlo approach associated with knowledge was first submitted to a conference [46], then to an international journal [50]. The work about the application of mathematical morphology to computer go was first published in a national conference [31], then in an international journal [48]. The work about spatial reasoning in go went through different seminars and conferences [3] before being published in a national journal [43]. Besides, the publication of the 70 page long paper [53], written with Tristan Cazenave, was both a long-term and large-scale work. The work performed from 1997 up to 2002 to achieve Indigo2002 is partly described in [49]. This is the reason why the five publications that illustrate our work at best have also been extensively published.

[50], “**Associating domain-dependent knowledge and Monte Carlo approaches within a go program**” 10 pages, forthcoming, Information Sciences, (2004). This paper shows how the Monte Carlo approach was associated with a knowledge-based approach in Indigo. Although this paper centers on a work carried out in 2003 only, it reflects at best the whole work performed since 1997 until now, in term of effective result achieved. In fact, the knowledge-based approach, adopted between 1997 and 2002, did not enable Indigo to achieve good results. The Monte Carlo approach, started in 2002, is promising, and gives good results when coupled with domain-dependent knowledge. Looking back on the whole 1997-2004 period, I consider the association of these two complementary approaches as a technical success. This ten-page paper was first published as a four-page abstract to the Joint Conference in Information Sciences (JCIS) conference. This paper is the second part of a three-paper serie about the application of Monte Carlo to go. It follows the previous work published with Bernard Helmstetter about the application of Monte Carlo alone [54], and it is followed by a work describing the association of Monte Carlo with global tree search, published recently [51].

[48] “**Mathematical morphology applied to computer go**” International Journal of Pattern Recognition and Artificial Intelligence (IJPRAI), vol. 17 n2, pp. 257-268, (2003). This paper shows how mathematical morphology is applied to recognize territory and influence in go. It describes operators that are similar to morphological dilation and morphological erosion, adapted to the game of go. This work has been used first in Indigo, and second in GNU Go. The model described in this paper was uncovered before 1995 while I was working on my PhD. The implementation described in the paper was performed between 1997 and 2000 when working on the time optimisations of Indigo. Thus, this paper also reflects our long-term work.

[53] “**Computer Go : an AI oriented Survey**” with T. Cazenave, Artificial Intelligence Journal, vol. 132, n1, pp. 39-103, (2001). This paper describes an AI oriented survey of computer go, which had never been done before. Now, it is often cited in papers about computer games. In some aspects of AI, the descriptions provided are based on our own experiments while developing GoLois and Indigo. Concerning the other aspects, the descriptions are based on the state of the art of computer go. Thus, this paper enables AI researchers to know about the techniques currently used in computer go.

[43] “**Les concepts spatiaux dans la programmation du go**” Revue d’Intelligence Artificielle, vol. 15, n2, (2001), pp.143-172. This paper illustrates the spatial reasoning techniques used in Indigo. It highlights the evaluation function of a go program that uses many spatial concepts such as connection, dividing, circling, territory, and influence. The concepts are recognized by using various tools such as mathematical morphology of the Hausdorff distance.

[49] “**The move-decision strategy of Indigo**” International Computer Game Association Journal (ICGAJ), vol. 26, n1, pp.14-27, (2003). This paper describes the move-decision strategy used by Indigo until 2002. This strategy was mainly based on two methods. The “calm” method performs very selective global quiescence tree search. The “urgent” method selects urgent moves by using domain-dependent heuristics. The urgent moves are checked with the very selective global quiescence tree search. Besides, other methods such as mono-color tree search, life and death tree search are presented in this paper.

4.2 Classification

This subsection classifies the whole set of publications into subsets, along the following lines: workshop, conference or journal, national or international event, with or without a reviewing process.

- 4 International journals with a reviewing process: [48, 49, 50, 53].
- 2 National journals with a reviewing process: [43, 47].
- 7 International conferences with a reviewing process: [36, 39, 45, 46, 52, 51, 54].
- 6 International workshops with a reviewing process: [30, 33, 35, 38, 40, 42].
- 5 National conferences with a reviewing process: [28, 29, 31, 34, 37]
- 2 National journals with no reviewing process: [14, 55].

5 Thematic presentation

Computer go is a very appropriate domain to perform AI experiments [53, 55], and AI community is made up of various domains. Thus, showing the interaction between our research and these domains is worth considering. Subsection 5.1 shows the link with cognitive science and DAI. Then, the following subsections describe other domains still used in the current program: uncertainty management and combinatorial games (subsection 5.2), retrograde analysis and tree search (subsection 5.3), spatial reasoning and evaluation function (subsection 5.4). The use of Monte Carlo techniques is examined by subsection 5.5.

5.1 Cognitive science and DAI

Our PhD work [32] was part of a cognitive science perspective. As explained in subsection 2.4, Indigo’s model was both computational and cognitive. In my PhD, the computational model was linked to the cognitive model. The working hypothesis gave a correspondence between a computational concept and a human concept. When the human player’s verbalizations contained terms referring to computational concepts, the cognitive concept corresponding to the term was termed “conscious” and the other ones were termed “not conscious” or “implicit” [34].

A go position can be broken down into strongly interacting sub-positions, particularly when two neighboring opponent groups are fighting to kill each

other. During such a fight, the group that gets the most liberties, or the most eyes, generally wins the fight. Practically, it is useful to manage an object “interaction” gathering the features of the fight between the two groups: eye number, liberties in common, proper liberties, winner of the fight, and so on. The two papers [37, 39] underline a correspondence between a “group”, in the go meaning, with an “agent”, in the DAI meaning. Thus, the example of go illustrates once more the central place occupied by interaction in DAI.

5.2 Sum of games, uncertainty representation

Since a go position is complex but structured, an interesting possibility consists in breaking down the whole position into sub-positions, then studying the sub-positions separately, and finally computing the global evaluation of the position by using the sub-position results. When the sub-positions are independent, Conway’s combinatorial game theory applies [76, 20], and the decomposition approach is relevant. Mathematical go [17, 21] and decomposition search [107] highlight this possibility.

Basically, the sub-positions in go are inter-dependent one of each other. However, although this is not theoretically correct, breaking down the position into sub-positions reduces the complexity of the problem, and this approach enables the programmer to start his study. Practically, the decomposition may be based on the recognition of groups and territories by using domain-dependent knowledge [32]. Then, the sub-positions have to be described in a useful manner, finally constructing the whole evaluation with these local descriptions. [40] underlines the sub-game description and [36] highlights the construction of the evaluation function with the local results.

5.2.1 Evaluation function including uncertainty

[36] was written when Indigo was an expert system evaluating moves with rules. Indigo made mistakes due to the lack of verification of his moves. But it could see its mistakes at the next move. Thus, many of these mistakes could be avoided by using a depth-one global tree search, calling a global evaluation function. The main problem was to take into account the uncertainty on the issue of fights between groups. In [36], for each “uncertain” group, i.e. neither dead nor alive, an uncertainty was defined by the difference between the position evaluation assuming the group is dead, and the position evaluation assuming the group is alive. When two groups are fighting, the closer the issue, the greater the risk. These principles presented in [36] are still used in the current release of Indigo.

5.2.2 Strength of groups with two numbers: P and Q

[40] studies the break-down of the global game into life and death sub-games. Moreover, it defines the importance of the “strength” of groups with two numbers P and Q. P is the number of iminical moves played in a row to kill the group, and Q is the number of friendly moves played in a row to make the

group alive. This approach has been compared to Tristan Cazenave's description of games [66] and to the "Possible Omission Number" (PON) of Tajima and Sanechika [141].

5.3 Tree search

This subsection shows the research related to tree search which is the prevailing paradigm in computer games. Until now, the relationship between tree search and computer go has been very special. Contrary to other games in which tree search is global, and calls an evaluation function at leaf nodes, the tree searches performed by go programs are mostly local, and few are global. Furthermore, in go the local tree searches are performed to compute the global evaluation. Thus, unlike in chess or othello programs, tree search *is used by* the evaluation function. However, some go programs, Indigo included, perform global tree search, and with the ever increasing power of computers, tree search will be used more frequently in the future. Thus, tree search is going to become a more and more useful domain to computer go.

5.3.1 Related work

Numerous publications deal with alfa-beta both in practice [62, 102] and in theory [92, 114]. Junghanns offers a recent overview of alfa-beta works [89]. This subsection is divided into four paragraphs: alfa-beta enhancements, best-first algorithms, algorithms including statistics or probabilities, and finally tree search applied to computer go.

Alfa-beta enhancements Alfa-beta [102] does not cost much memory but uses a time exponential in the search depth. Without its enhancements, its efficiency is poor. The two basic enhancements are the transposition table [88] and iterative deepening [135, 93]. Other enhancements such as null move [78] or history heuristic [125], quiescence search [16], singular extensions [11], Scout [113], MTDF [116] are also worth considering.

Best-first algorithms. Other algorithms expand the nodes in a best-first manner. The downside is to keep the whole tree in memory. SSS* by Stockman [138] associates two techniques, min-max and A*, in a clever way. Proof-number search by Allis [10] uses proof numbers to determine which node to expand. B* by Berliner [22] uses an optimistic evaluation and a pessimistic one. Korf and Chickering's best-first algorithm simply expands the leaf node situated on the principal variation [94]. To expand the next node, conspiracy number search analyses which nodes may change the value of the root if their own value changes [104, 126].

Using statistics and bayesian back-up rule Rivest shows that many back-up rules can be used during tree search [121]. When the evaluation does not correspond to a value but to a probability distribution, Palay [112] defines

the bayesian back-up rule that is used by Baum and Smith [15]. This approach can be linked to the Monte Carlo simulation approach in which the evaluation is an average value on a sample of values. Logistello by Michael Buro was built by using statistics. The evaluation function was obtained through a linear regression, and the tree search algorithm uses the probCut heuristic [60]. ProbCut performs a shallow tree search to obtain approximate values of the alfa-beta window. This enables the program to prune some moves at a low cost with a given level of confidence. Sadikov, Kononenko et Bratko [122] describe a practical experiment performed on KRK chess endgames to show the link between the error due to tree search when using an erroneous evaluation function. This example shows that tree search introduces a bias in mini-max values that depends on the search depth, but this bias does not alter the playing level of the program using this tree search.

Tree search applied to computer go In go, Ken Chen studied the effect of an error in the evaluation function and the effect of selectivity on both mini-max values of the root, and the playing level of the program [72]. Algorithms specific to tactical sub-games of go were uncovered recently: Lambda-search by Thomas Thomsen [146], and Abstract Proof Search and derived algorithms by Tristan Cazenave [67, 68, 69]. 4x4 go was solved in 2000 by Sei and Kawashima [131], and 5x5 go by Erik van der Werf in 2002 [154]. We performed tree search experiments on very small boards [41]. Ken Chen also studied the opportunity to use global tree search in go [71]. Decomposition search which was published by Martin Müller [107] uses the sum-of-game paradigm described in [108].

5.3.2 Tree search in Indigo

In this subsection, our work is described in connection with tree search: the various kinds of tree search used in Indigo until 2002 [49], tree search on very small boards [41], retrograde analysis [42], and the association between selective global tree search and Monte Carlo in 9x9 go.

Tree search until 2002 Until 2002, Indigo used numerous tree search techniques [49]. To read ladders at the tactical level, tree search gives correct results [30]. On non-circled groups, at the life and death level, tree search has to be very selective and the depth must be very limited. At the global level, Indigo2002 used selective global quiescence tree search. Since the evaluation function contains errors, global quiescence tree search was used in a limited way.

Tree search and retrograde analysis on small boards Retrograde analysis [147, 148] being used with success in chess, we tried this technique on small go boards in 2001 [42]. This technique is interesting for its automatic feature. We also carried out small board experiments in order to master classical alfa-beta enhancements [41]: transpositions [88], iterative deepening [135, 93], history heuristic [125], null move [78], and MTDF [116].

Tree search associated with Monte Carlo in 2003 Indigo2003 is based on three modules: a knowledge-based move generation and evaluation function module, a Monte Carlo module, and a global tree search module. The association between them is described in [51]. The global tree search is similar to iterative deepening. At a given depth, random games are processed at leaf nodes, and the branches of the tree are pruned as long as the random games are performed. The algorithm stops either when the maximal depth is reached, or when there remains one single move at root node.

5.4 Evaluation functions and spatial reasoning

Evaluating non terminal positions constitutes the major difficulty of computer go. As shown by Patrick Ricaud in his PhD thesis [119, 120], abstraction is necessary to build an evaluation function. Our research on this topic has used two key ideas: territory and influence recognition with mathematical morphology [48, 31] and inimical interaction [37, 39]. A go position contains much information classified into different categories. As a result it is hard to reduce this information to a single value. [43] shows the work carried out within such a context. [45] assessed the effect of using two mathematical tools in evaluation functions: distance and dimension.

5.4.1 Mathematical morphology and inimical interaction: two keys

Firstly, in my PhD, mathematical morphology developed by Jean Serra [132] was used to recognize “groups”, “territories” and “influence” in a go position [48, 31]. The groups are equal to the morphological closing of the stones, the territories are equal to the groups minus the stones, and the influence corresponds to the morphological dilation of the stones, minus the territories. When compared to human capabilities, this yields very good results. Indigo and GNU Go use this model. Secondly, the two opponent group comparison principle was defined, and named “interaction”, which is very important in go. During a fight between two groups, the group with the most liberties or the most eyes, generally wins the fight. In practice, the interaction is the object that contains this information. [37, 39] are two papers highlighting the interaction notion in go. They were already mentioned in subsection 5.1. Finally, the set of morphological and interactive concepts lead to the Indigo conceptual evaluation function, which is still used in the current release. Coupled with the global move generation module, it provides the moves to the Monte Carlo module.

5.4.2 Spatial concepts and evaluation function

The upside of working on a complex evaluation function in a visual domain such as go is to be a test domain for the spatial reasoning researches, that have been pursued for a long time [95, 79, 75]. The evaluation function is one of the major goals in the Indigo project, and an important effort has been made in this direction. This effort allowed to uncover, classify, use and test spatial

concepts [43], either metric or dimensional [45]. [43] starts from the difficulty of computer go to build a complex evaluation function, whose nature is strongly spatial. Then, it is interesting to describe to the spatial reasoning community the spatial concepts used by a go program. The concepts are territory, connection, circling and dividing. This work partly results from the participation in several working groups on spatial reasoning, in which it was presented: GDR I3 group [1] about spatio-temporal model in 1998, “Quando et Urbi” group [151] set up by Gérard Ligozat and Jean-Paul Sansonet, and finally the “Journées Nationales sur les Modèles de Raisonnement” (JNMR) 1999 [3], organized by GDR I3 as well.

5.4.3 Metric and dimensional concepts in evaluation functions

[45] studies the results brought about by tools based on distance and dimension notions. These tools are used by go programs playing on very small boards (4x4), and they correspond to evaluation functions. The distance-based evaluation function underlines the player’s will to minimize the distance between his stones, and maximize the distance between his opponent’s stones. For each color, the dimension-based evaluation function reflects the number of groups, the size of these groups, and the size of the whole set of this color. These tools were evaluated experimentally through tournaments organized between go programs using different values of parameters. The method for obtaining the best go programs was inspired from genetic algorithms. After a tournament, the best programs were kept as such for the next tournament, other programs were kept with a mutation, and the worst programs were replaced by new ones, while keeping an adequate balance between exploration and exploitation. In the end, the population of programs converged to an “optimal” program whose values enable us to assess the relative merits of each notion, distance and dimension.

5.5 Monte Carlo

Using statistics to obtain an evaluation function is a very good idea developed by Abramson [6], and by Brügmann in his simulated annealing approach [57]. Bernard Helmstetter updated this idea when he started his PhD thesis under the direction of Tristan Cazenave. This subsection describes the work performed on Monte Carlo within the Indigo project. First, Monte Carlo alone is described [54] in subsection 5.5.1. Second, the association of Monte Carlo and knowledge is underlined [46, 50] in subsection 5.5.2. Third, the association of global tree search and Monte Carlo is highlighted [51] in subsection 5.5.3. Finally, in subsection 5.5.4, we sum up the reasons why, to us, Monte Carlo is promising.

5.5.1 Monte Carlo with very little knowledge

The research described in [54] is derived from Abramson’s idea [6], and Brügmann’s [57]. In Abramson, the idea consists in computing the evaluation of a given position by performing random games starting on this position. Each random game

terminates on a final position which is easy to evaluate. Then, the evaluation of the given position is the average of the sampled final evaluations. It is also possible to perform depth-N tree search. In Brügmann, the idea corresponds to simulated annealing. Bernard Helmstetter followed this line while developing Oleg. As for me, I developed Olga which uses progressive pruning [54]. The idea of simulations is very common in games containing random such as backgammon [144, 145] or hidden information such as poker [24] and scrabble [134]. As a result, Oleg and Olga, using very little knowledge, were ranked on a par with Indigo2002, heavily knowledge-based program, when playing on 9x9 boards, which was most amazing. In [54], we assessed several enhancements such as progressive pruning, the all-moves-as-first heuristic, simulated annealing, and depth-2 tree search. Besides, a Monte Carlo go project was supervised in cooperation with Gilles Zémor at ENST (Ecole Nationale Supérieure des Télécommunications) [64].

5.5.2 Monte Carlo and knowledge

By observing that the statistical approach and the knowledge-based approach gave similar playing levels on 9x9 boards, it was natural to measure the playing level of a program using both approaches. We issued three releases playing against each other: the knowledge-based release, the statistic-based release, and the knowledge-and-statistic-based release. Quantitative results are provided in [46]. They show that the association of Monte Carlo and knowledge is very effective. The association is made up of two parts. First, the program uses the knowledge-based move generator to select few moves which are inputs of the Monte Carlo module. This way, tactically bad moves are eliminated and the program plays faster than before. Second, random game move generation uses small patterns to read the move urgencies in a table. This way the random games come closer to go games than before, and the evaluations obtained are more significant. Consequently, the playing level is increased. This work is to be published in Information Sciences [50] in 2004.

5.5.3 Monte Carlo and selective global tree search

In order to prepare Indigo for the 9x9 go competition of the computer olympiads held in November 2003 in Graz, an algorithm associating Monte Carlo and global tree search was designed [51]. Its content was also mentioned in subsection 5.3.2.

5.5.4 Why Monte Carlo ?

Why do we think that Monte Carlo is so promising ? The two main difficulties of computer go are: the difficulty to build a good evaluation function on non terminal positions, and the combinatorial complexity of a global tree search. Let B be the branching factor and L the length of games. The game tree size is in B^L . To some extent, Monte Carlo reduces these two difficulties. First, [6] yields a very costly but very robust statistical evaluation function. The

statistical searches are based on terminal position evaluations. In this view, although the path to reach these positions is random, these searches are reliable. Besides, terminal evaluations are obvious and fast to compute. Second, let N be the random game number, the complexity of global statistical search is in $N \times B \times L$, thus significantly inferior to B^L . The computing power of current computers make the Monte Carlo approach possible. To sum up, Monte Carlo reduces the two obstacles of computer go by offering a statistical method to evaluate non terminal positions, and a global statistical search providing the selected move with a confidence level. Of course, Monte Carlo does not solve the local tactical problem which cannot be solved without tree search.

6 Computer games

This section presents our research within the general context of computer games. Subsection 6.1 deals with computer go. Subsection 6.2 describes computer games for (classical) games using global tree search. Subsection 6.3 mentions other games: combinatorial games and games containing randomness or hidden information. Finally, subsection 6.4 underlines computer hex and computer amazons for which we have supervised students.

6.1 Computer go

This subsection yields a very selective computer go state of the art, then it shows the contribution of the Indigo project to this domain.

6.1.1 Related work

This part shows some viewpoints on computer go: the general work, the combinatorial complexity, the best programs, the small boards, Albert Zobrist's contribution, life and death problems, and neural networks.

General works The most two recent publications on computer go are an AI oriented survey [53], and a classical state of the art [109] by Martin Müller. Ken Chen classifies go programs according to the decision process used: global tree search, temperature-based, expert system or other. [91] is a general paper dealing with the engineering of go software. Markus Enzenberger regularly updates an online bibliography [80].

Combinatorial complexity Lichtenstein and Sipser showed that go is P-space hard in the board size [97]. Fraenkel and Lichtenstein showed that finding out a perfect strategy in $N \times N$ board games with complete information is time exponential in N .

Best programs The best program in 1990, Goliath written by Mark Boon, is described in his dissertation [27]. The pattern-matching algorithm of Goliath (also used in Indigo) is described in [26]. Go++ [117], the best current program by Michael Reiss [118], is not described anywhere except in [56]. Go Intellect by Ken Chen, was world champion in 1994. Its move decision process is described in [70]. [74] contains a set of heuristics dealing with life and death, and used by Go Intellect and Goemate. Many Faces of Go [84] by David Fotland is one of the best programs. A long time ago, David Fotland published an algorithm computing ladders [85], and recently a set of heuristics dealing with eyes [86], used by Many Faces of Go. KCC Igo, the North-Korean program, Goemate, the Chinese program by professor Chen, Wulu, a program derived from Handtalk, Haruka, a Japanese program by Ruichi Kawa, are commercial programs that attend most of the international competitions successfully. Unfortunately, they are not described anywhere. GNU Go is the program of the Free Software Foundation. It has almost reached the level of the best programs. Programmers from all the world are working on it under the supervision of Daniel Bump. All these programs have achieved a level which is difficult to assess with accuracy. Basically, the first games against them could give the impression they are about 8th kyu². But actually they are not: the following games, uncovering their weaknesses, show they can only be 15th kyu.

Small boards Thorpe and Walden first studied go on small boards [149, 150]. Tree search enables computer go to solve 4x4 go [131], and recently 5x5 go [154].

Albert Zobrist Albert Zobrist is famous for two contributions: the influence model for computer go [165], and the hashing model for computer games [166] which are still used in go programs.

Life and death Life and death problems (tsumego) are the subjects of active research. Benson's algorithm computes life and death statically. It can be used within tree search. GoTools, a program by Thomas Wolf, solves and generates very high level life and death problems [161, 162, 163].

Neural networks Neural networks [25] are applied to computer go. In 1994, Schraudolf, Dayan and Sejnowski used a network on a raw board [130]. Markus Enzenberger's network used a priori knowledge, which gave NeuroGo [81, 82]. [83] is the following work including a connection algorithm within the network. Honte, by Fredrik Dahl, uses neural networks at several abstraction levels: to generate moves, to compute life and death on groups, and to recognize territories [77].

²A beginner is 30th kyu, an average player is 10th kyu, a strong player is 1st kyu or 1st dan, and the best professional players are 9th dan

6.1.2 Indigo project's contribution

Indigo project's contribution is can be seen in numerous publications and in its participations in international computer competitions with ever-improving results.

Publications since 1995 In the publications since 1995, [53] has very often been cited. The publications on Monte Carlo [54, 46, 50] have drawn go programmers' attention. [31, 48] have already been used. Jérôme Dumonteil participated in GNU Go development, and convinced the team to use our "5-21 algorithm". Descriptions of internal details of Indigo are available [32, 30, 44, 49]. Furthermore, specific papers describe the use of specific techniques in go. Retrograde analysis [42], incrementality [38], uncertainty representation [36], go sub-games [40] are approaches interesting for go programmers.

Competitions since 1998 The second aspect of Indigo project's contribution lies in the confrontation. At first glance, attending a tournament consists in a confrontation set, but in fact, it can conversely be seen as a collaboration set. Basically, in the long term, a game can be studied offline, and positive and negative conclusions can be drawn from this game. Since 1998, Indigo has participated in several computer go competitions, listed below in chronologically converse order:

- 9x9 Computer Olympiads, Graz, Autriche, November 2003 (4th/10),
- 19x19 Computer Olympiads, Graz, Autriche, November 2003 (5th/11),
- Computer Go Festival, Guyang, Chine, October 2002 (6th/10),
- 21st Century Cup 2002, Edmonton, Canada, July 2002 (10th/14),
- 13x13 Stefan Mertin's tournament 2002, (15th/19),
- 9x9 Stefan Mertin's tournament 2001, (7th/11),
- Mind Sport Olympiad 2000, London, August 2000, (5th/6),
- Ing Cup 1999 Shanghaiï, November 1999 (13th/16),
- Computer Go Forum 1999, Tokyo, July 1999 (24th/28),
- Ing Cup 1998 London, November 1998 (10th/17).

These results are set in an international context against the best go programs in the world. The last two results (Indigo was ranked twice in the first half of the rankings) obtained in Graz in 9x9 go [153] and 19x19 go [73] are very promising, and they should be confirmed in the future. The announced aim of producing a go program "as strong as possible" has been reached: Indigo2003 is about 12th or 13th kyu, i.e. 4 or 5 stones below the best programs.

During computer competitions, Indigo was repeatedly paired against programs of similar strength. It played against SmartGo [90] by Anders Kierulf, GoLois [65] by Tristan Cazenave, Explorer [106] by Martin Müller, NeuroGo [81] by Markus Enzenberger, Aya [164] by Hiroshi Yamashita, and consequently, the authors of the programs developed a mutual respect. Indigo played against stronger programs too: Go4++ [117] by Mick Reiss, Many Faces [84] by David Fotland, Goemate by Zhinxing Chen, GNU Go [58] by the FSF, GoAhead [160] by Peter Woitke, and Go Intellect by Ken Chen.

6.2 Classical game programming

This subsection shows the results obtained in classical games, i.e. games in which global tree search is the appropriate method to use.

6.2.1 Mind games in general

Games have been studied for the last sixty years: as early as 1940's Von Neumann and Morgenstern had studied min-max trees [156]. Recently, Schaeffer reviewed the breakthroughs in computer games [128]. Allis' thesis is very interesting to know the state of the art of many games [9]. In the special issue of AI Journal, Jonathan Schaeffer and Jaap van den Herik describe the state of the art for mind games [129]. Jaap van den Herik, Jos Uiterwijk and Jack van Rijswijck have assessed the likely improvements of the playing level for these games in the next ten years [152].

6.2.2 Chess, othello, checkers

Programming chess, othello and checkers mainly use tree search described in subsection 5.3. Claude Shannon's paper [133] was one of the first papers on computer chess. Recently, [61] has described the Deep Blue development five years after its win over Kasparov. But we may wonder where the playing level of the best chess programs ranks in 2004. On the one hand, Kasparov drew Fritz, one of the best chess programs. On the other hand, Fritz took first place tied with Shredder at the last world championships held in Graz in November 2003. Although Deep Blue was taken off the competitions, an increasing number of chess programs have reached the playing level of the best human players.

Programming othello was marked by Logistello developed by Michael Buro [59, 60]. In 1997, Logistello lay far beyond the best human playing level. Programming checkers started with the famous works by Samuel [123, 124]. Then, Jonathan Schaeffer wrote Chinook, whose history is described in [127]. The best checkers programs are now stronger than the best human players. However, Jonathan Schaeffer thinks that checkers is unlikely to be solved soon.

6.3 Other viewpoints on game programming

This subsection provides two other viewpoints on computer games: combinatorial game theory, and hidden information games.

6.3.1 Conway's combinatorial games

The two basic combinatorial game references are “On Numbers and Games” [76] by John Conway, and “Winning Ways” by Berlekamp, Conway and Guy [20]. In “What is a game?” [87], Fraenkel defines a game starting with game 0 created on day 0, then continuing with the games *, 1 and -1, created on day 1 with game 0, and the right to play one move. On day N, the games that reach a game created on day N-1 in one move are created. In [17], Elwyn Berlekamp introduces mathematical go, and [21] is a more detailed description of this theory. [110], by Martin Müller and Ralph Gasser, provides an overview of the combinatorial games' relevance to go endgames. Howard Landman describes go eyes by using the combinatorial games' formalism [96]. Bill Spight comments on a game in which Jiang and Rui, two professional players, get the right either to play a normal move on the board or take a ticket of points [137]. The tickets have an integer value decreasing from N points down to 1 point. Thus, the endgame is played out almost normally, but sometimes a player chooses to take a ticket instead of playing a move on the board. Besides, [19] describes a game made up of four sub-games: one chess game, one go game, one domineering game, plus one checkers game. In its turn, the player chooses one of the four sub-games, in which he plays his move. The game described in the paper is played out by applying the combinatorial game theory, demonstrating its unifying power.

To sum up, combinatorial games offer a formal framework useful to describe game aspects of go, particularly relevant when the architecture of the go program breaks down the global game into sub-games.

6.3.2 Games containing randomness or hidden information

Game programming in games containing random, such as backgammon, or hidden information (poker, bridge, scrabble) includes simulations in a very natural way. The random function of the computer is useful to simulate dice rolls or hidden cards. [24] in poker, [144] in backgammon, and [134] in scrabble, show the effectiveness of simulations. Although go is a complete information game, simulations can be used in go too. We think that when two players play a complete information game against each other, they use their private conceptual description of the board. In this respect, some conceptual information remains hidden. Thus, we think that using simulations is also adapted to games that usually belong to the complete information game category.

6.4 Computer game project management

This subsection presents our experience of supervising students on projects or internships concerning two mind games: amazons and hex.

6.4.1 Amazons

Amazons [142] shares two aspects common to chess and go: the players move amazons like queens in chess, and they occupy territories like in go. Müller

and Tegos described relevant objects and methods for playing amazons [111]. Snatzke studied the effect of tree search on small boards [136]. The evaluation function of Amazong [98], the best current amazons program, by Jens Lieberum, is described in [99]. Besides, amazons has similarities with the angel game defined by Berlekamp [18]. A finite size variant of the angel’s problem is DukeGo [103].

An amazons move can be broken down into two sub-moves: first, move the amazon, then throw the arrow. As a result, there are two ways to perform random games, and two ways to handle the Monte Carlo method. Concerning random generation, the uniform probability method is slow but to some extent, correct. But another method can be designed. Let us call it the "fast" method. In the slow method, the move generation makes necessary to move the amazons in order to determine where to throw the arrows, and then *to move back* the amazons, which can be avoided with the "fast" method. In the fast method, the amazon move is first chosen randomly and performed, then the destination of the arrow is chosen randomly and performed. This method is simpler and faster than the slow method (the amazons do not move back), but it does not correspond to the uniform probability. Thus, to some extent, the fast method is not correct.

Moreover, the Monte Carlo method can be split into two stages as well. Firstly, assuming that an amazon moves to a destination, compute the mean result on random games. Then choose the best amazon and the best destination, and perform the move on the board. Secondly, assuming that an arrow is thrown to a destination, compute the mean result on random games. Then choose the best destination, and throw the arrow on it. This method is close to a depth-two tree search in which the friendly player moves twice. This approach largely diminishes the number of random games to perform. Unfortunately, what you gain in time is lost in playing level.

In 2003 and 2004, I supervised students on Monte Carlo amazons. In the summer 2003, Arnaud Caillieretz developed a first release of an amazons program. In the fall 2003, I gave a project to Master’s students whose first qualifications were in statistics and not in programming (MST ISASH2). They had to produce a Monte Carlo amazon program by testing the possibilities described above. Technical reports are available [63, 23, 101, 7]. Meanwhile, I submitted the same subject to other Master’s students whose first qualifications were in programming, and not in statistics (MIAIF2). They also obtained Monte Carlo amazons programs, and their programs were faster. But, due to different goals, they did not have to provide the statistical results of these experiments. Supervising those two kinds of students was a very fruitful experience.

6.4.2 Hex

Hex [12] is a connection game played on a board in which points have six neighbors. Jack van Rijswijk described his program Queenbee that uses an interesting evaluation function [155]. Hexy is the best current program. It is based on virtual connections [13]. In Hex, I supervised a student that was very fond

of genetic algorithms. His work lasted four months. While I helped him to formalize his ideas about using genetic algorithms, Hex is a very tactical game and I had him use tree search [8]. He learnt a lot during that internship, which was a positive point. His program was weaker than Hexy or Queenbee, but was playing reasonably well, which was the goal I wanted him to reach.

7 Other research activities

This section provides a brief description of other research activities.

7.1 Workshops

In addition to presentation in conferences, I have regularly presented my work in seminars or scientific meetings, and once I chaired a special session in a conference for young researchers.

- GDR I3 [1], Modèles spatio-temporels: “Programmation du go et raisonnement spatial”, December 1998,
- Group “Quando et Urbi” [151], directed by Gérard Ligozat and Jean-Paul Sansonnet, same title, March 1999,
- “Journées Nationales sur les Modèles de raisonnement”, JNMR99 [3], same title, March 1999,
- Scientific meeting of the computer science and mathematics department of the University of Paris 5, “Le projet Indigo”, March 2000,
- Chairman of the “Art et Jeu” session in “CJC-03” (Colloque Jeunes Chercheurs en Sciences Cognitives), June 2003,
- Journée CRIP5, “Le projet Indigo”, January 2004,
- Group “IA et jeu” under the supervision of Vincent Corruble at LIP6, “Le projet Indigo”, February 2004.

7.2 Other contributions to academic research

Within the academic research, I participate in underlying activities such as reviewing papers or conference organization. Here is the list of my personal contributions:

- Conference “Computational Intelligence in Games”, CIG-05 [100]: program committee member,
- Conferences “Computers and Games”, ACG-10, ECAI-2000, and ICGA Journal: reviewer,
- Régis Monneret’s Ph. D thesis [105], July 2000: examiner.

Over the last few years, the conference calendar has been planned to answer the growing interest for computer games. In 1998, the conference “Computers and Games” (CG) was set up and is to take place every two years, on even years. The conference “Advance in Computer Games” (ACG) is still a great success, and takes place every four years, the last one being in 2003. Almost each year a game programming workshop is organized in Japan. Besides, when they are not already associated to a CG or ACG conference, the computer olympiads are associated to a workshop organized by the computer science department of the Maastricht university. CG or ACG conferences are frank successes but they do not cover all the year calendar. In view of the researchers’ interest for computer games, a new conference has now been organized for April 2005: ”Computational Intelligence for Games” (CIG). I accepted to participate in this conference as a member of the program committee.

7.3 Supervising students

This part lists the internships I have been responsible since 1992.

Pierre Recoque ”Pattern-matching au jeu de Go” (DEA IARFA, 1992). This internship co-directed with Jacques Pitrat enabled Pierre Recoque to develop a pattern-matching algorithm specific to the game of go.

Tristan Cazenave ”Création automatique de règles à patterns” (DEA IARFA, 1993). This internship co-directed with Jacques Pitrat enabled Tristan Cazenave to develop a first tool that automatically generates patterns, which was useful to start his PhD thesis.

Vincent Airault [8] ”De l’application de méthodes classiques au jeu d’Hex vers une utilisation des algorithmes génétiques” (MIAIF2, 2001). While the initial aim of Vincent Airault was to apply genetic algorithm to the game of hex, this internship enabled him to master the practice of tree search techniques, and to obtain an early hex program. Obtaining a hex program by using genetic algorithm remains an interesting perspective.

Jean-François Goudou, Vincent Castillo ”Monte Carlo Go” (ENST, Spring 2003) [64]. This project co-directed with Gilles Zémor consisted in reproducing the Monte Carlo go experiment, which was performed with success.

Arnaud Caillieretz ”Monte Carlo Amazons” (IST, Paris 6, Summer 2003). This internship was a first attempt to apply Monte Carlo to amazons. This experiment was useful to manage the projects about programming and statistics for MST ISASH2 and MIAIF2 in the fall 2003.

Christophe Truong "Monte Carlo chess" (IST, Paris 6, Summer 2003).). This internship was a first attempt to apply Monte Carlo to chess. Although the work was well-done, chess being a very tactical game, this technique was not conclusive.

8 Perspectives

The Indigo project offers numerous perspectives: the Monte Carlo continuations (in go or in amazons), the automatic generation of pattern databases, reinforcement learning, and parallelism.

8.1 Automatic generation of patterns within a bayesian approach

The first perspective is to replace the hand-built pattern databases by automatically constructed databases. This work has already started. The approach consists in using bayesian learning of K-nearest patterns [25] extracted from recorded games available on the Internet.

The global move generator module uses three pattern databases answering distinct requirements. The first database is made up of patterns used by the conceptual evaluation function. The second one contains simple josekis³, and the third one is composed of edge patterns. The patterns contained in these three databases are general and scarce. The guideline is to replace these three databases by a single one using the K-nearest neighbor formalism. Supervised learning will be performed on the examples of positions belonging to professional games. Computing urgencies will use Bayes' formula. For each point and each pattern, the proportion of moves, played or not, and the proportion of matches will be counted.

8.2 Continuing Monte Carlo go

Given the promising results obtained by Monte Carlo in 2003, the most natural perspective is to pursue with this approach. Today, the 3x3 pattern database used by random games has been built in two ways. The database built in the first way was obtained by intersecting the available pattern database used by the evaluation function of Indigo with 3x3 windows. We called this way, the "manual" way, because it uses a database built by hand over the years. The database built in the second way was constructed with the bayesian approach. Unfortunately, the number of available professional games (360) was too low to compute significant probabilities. Thus, we aim at reproducing this experiment with many more professional games.

A third direction rests on reinforcement learning [140] and Q-learning by Watkins [157, 158], to improve the accuracy on the pattern urgencies automatically. In addition to its autonomy and independence of extraction from

³move sequence played out in a corner at the beginning of a game

professional games, this approach offers the advantage of being adapted to the problem: finding out action values in the meaning of [140]. The aim is not to find the correct pattern to be selected in a given position, which is done through supervised learning. While the Monte Carlo episodes give good results when they correspond to full games, it may be interesting to perform shorter episodes. Tesauro and Galperin [145] performed such an experiment in backgammon.

8.3 Parallelism

When the speed is crucial, studying parallel approach seems relevant. For example, the Monte Carlo experiments carried out by Tesauro et Galperin used parallelism [145].

8.4 Applying temporal difference method in the game of go

Temporal difference algorithm [139] by Richard Sutton was applied successfully by Tesauro to backgammon [144, 143]. In go, Dayan and Sejnowski applied it without any go knowledge [130]. Then, Markus Enzenberger used it, by adding go knowledge, which resulted in NeuroGo [82]. This approach looks promising. However, applying temporal difference algorithm also means looking for a good evaluation function automatically. But, to some extent, Monte Carlo has already answered this requirement. If developed, this approach must be compared to the existing evaluation functions we already own: the conceptual evaluation function of Indigo, and the Monte Carlo evaluation function.

8.5 Continuing Monte Carlo amazons

In 2003, Arnaud Caillieretz, a student from IST at Paris 6 university, carried out a work on Monte Carlo amazons. In the teaching unit “Programming and Statistics” for MST ISASH, and for MIAIF, Monte Carlo amazons projects were supervised. I want to pursue this supervising activity in Monte Carlo amazons. In go, the Monte Carlo approach occurs after the knowledge approach. In Amazons, this would be the contrary because the game is new and the knowledge about this game is still in its early stages. A collaboration with Jens Lieberum, author of Amazon [99], the best program, is a perspective that we hope to make effective.

9 Conclusion

This document has presented computer go researches performed since 1997 within the SBC group of the IAA team, directed by Dominique Pastre, within the CRIP5 laboratory at the University of Paris 5.

Computer go is an interesting topic for AI, our aim is to obtain a go program as strong as possible while participating to competitions and publishing our

results. We showed a chronological overview of the Indigo project highlighting our research method based on two principles: the practical principle and the didactical one. The practical principle consists in keeping ideas that works in practice and removing the other ones. The didactical one consists in regularly varying the techniques used. This is likely to uncover interesting elements. The didactical principle also enables us to acquire new knowledge in a new domain. Over the period 1997-2002, we developed Indigo by using a knowledge approach. The move decision strategy used in Indigo2002 is provided by [49]. In 2003, we set up a new version of Indigo using both the Monte Carlo and the knowledge approach. Indigo ranked in the first half of the two go competitions organized on 9x9 and 19x19 boards at the computer olympiad held in Graz.

Then, various viewpoints on this work were given, in terms of AI and computer game sub-domains. [53] written with Tristan Cazenave is now cited in many computer games publications. [48] describes the application of mathematical morphology to computer go that brought about the 5-21 algorithm, referred to within the computer go community and used by Indigo and GNU Go. Spatial reasoning take an important role in go, and is closely linked with the difficulty of building a good evaluation function [43, 45]. The Monte Carlo approach was the richest viewpoint. It is described under three viewpoints: Monte Carlo alone [54], Monte Carlo with knowledge [46, 50], and Monte Carlo with selective global tree search [51], which gave good results in computer go competitions. The more powerful the computer gets, the better this approach works.

Future researches will still use statistical approach to improve Indigo: bayesian learning [25] or reinforcement learning [140]. Parallelism, temporal difference method and computer amazons are other perspectives worth considering, because they yield other valuable results to the computer game and AI communities.

10 Acknowledgements

I thank Dominique Pastre, professor in computer science at the University of Paris 5 , for warmly welcoming me in the SBC team in 1997, and for having carefully followed my researches. Although my research subjects are different from her's (automation of mathematical reasoning), our approaches are similar: solving a complex problem.

I thank Patrick Greussay, professor in computer science at the University of Paris 8 , for his interest in computer go and his warm welcome to his AI laboratory in the spring 1997.

I thank Jacques Pitrat, research director in computer science emeritus at the CNRS, head of the “Métaconnaissances” team in the LIP6 laboratory at the University of Paris 6 , for directing my PhD thesis, and for his interest in computer games, one of the best application domains of AI.

I thank Bernard Victorri, research director in computer science at the CNRS in the LATTICE laboratory associated to the “Ecole Normale Supérieure”, for

his encouraging attitude regarding my works, and more generally to computer go.

I thank Gérard Ligozat, professor in computer science at the University of Paris 11 at LIMSI laboratory, and Mario Borillo, research director in computer science at the CNRS in the IRIT laboratory, for their critical insights into the links between computer go and spatial reasoning.

I thank Jean Michel, research director in mathematics at the CNRS, one of the best French go players, for his passion for go programs on which he offers his critical view.

I thank Jean Serra, research director at the “Ecole des Mines de Paris”, director of the mathematical morphology center, and co-author of mathematical morphology.

I thank Ken Chen, professor in computer science at the University of North Carolina, author of Go Intellect, responsible for the Heuristic Search and Game Playing session at the JCIS conference, for his competence and his positive comments on my work.

I thank Martin Müller, associate professor in computer science at the University of Alberta, author of the reference go program Explorer. His enlightening ideas and his work on computer go provide me with an enriching example in terms of combinatorial game theory.

I thank Michael Buro, associate professor in computer science at the University of Alberta, author of the best othello program Logistello, for his ideas, his helpful discussions and his clever vision on computer games.

I thank Jaap van den Herik, professor in computer science at the University of Maastricht, IKAT (Institute for Knowledge and Agent Technology) director, editor-in-chief of the ICGA Journal, Jonathan Schaeffer, professor in computer science at the University of Alberta, director of the GAMES group, and Hiroyuki Iida, professor in computer science at the University of Shizuoka, for their academic framework which fostered a world wide recognition of computer games.

I thank Tristan Cazenave, associate professor in computer science at the University of Paris 8, accredited to supervise research, for his competence, his go program GoLois, and his tactics-oriented approach.

I thank Bernard Helmstetter, currently PhD student under the supervision of Tristan Cazenave at the University of Paris 8, for his view on computer go, his program Oleg, and his liking for the Monte Carlo approach. I also wish to congratulate him for his French 2003 go champion title.

I thank Anders Kierulf, Markus Enzenberger, Hiroshi Yamashita, Jimmy Lu, Sinichi Sei, Peter Woitke, David Fotland, Zhixing Chen, Mick Reiss, authors of SmartGo, NeuroGo, Aya, GoStar, Katsunari, GoAhead, Many Faces of Go, Goemate, Go4++, opponent programs of Indigo, either on the computer go ladder or during tournaments. I do appreciate their competence and the passion to computer go they share with me.

I thank all the members of the GNU Go team, directed by Daniel Bump. Thanks to GNU Go, computer go was released to the computer science community and to the go player community . I thank Jérôme Dumonteil for transferring

my PhD work about mathematical morphology to the GNU Go team.

I thank Jean-Charles Pomerol, professor in computer science at the University of Paris 6, director of the LAFORIA laboratory in 1997, editor of the RIA journal, and Jean-Marc Labat, professor in computer science at the University of Paris 5, director of the CRIP5 laboratory, in charge of the computer games chapter in the RIA journal.

I thank the mathematicians of the department, particularly Thierry Cabanal-Duvillard, associate professor, for his advice on statistics and probabilities.

I thank Sylvie Després, associate professor in computer science at the University of Paris 5, accredited to supervise research, and Marie-Jo Caraty, when she was associate professor in computer science at the University of Paris 6, and now professor at the IUT of Paris 5, for their support in some crucial moments.

I thank Martine Chauffeté, English teacher at the “Centre Technique de Langues” (CTL) of the University of Paris 5, for her precious help to improve some of my publications.

I thank Thierry Raedersdorff and Laurent Moineau, system engineers of the department, for their advice about Linux machines, used to perform experiments over nights and week-ends.

I thank my wife, Valérie, for her thoughtful and kind support, and my children Pablo, Lucas and Neïma, for their everyday cheerfulness.

Finally, I do thank my parents, André and Anne-Marie, for all they gave me.

References

- [1] Gdr i3: Information interaction intelligence. sis.univ-tln.fr/gdri3/.
- [2] International computer game association (icga). www.cs.unimaas.nl/icga/.
- [3] Journées nationales sur les modèles de raisonnement 99 (jnmr99). www.cril.univ-artois.fr/~marquis/JNMR99/articlesJNMR-99.html, Mars 1999.
- [4] 21st century cup. www.intelligentgo.org/en/igf/21cc2002/new-index.html, 2002.
- [5] Computer olympiad graz 2003. www.cs.unimaas.nl/Olympiad2003/, 2003.
- [6] B. Abramson. Expected-outcome : a general model of static evaluation. *IEEE Transactions on PAMI*, 12:182–193, 1990.
- [7] E. Agbodjan-Prince, Y. Aït Bachir, S. Bourham, S. Deleplace, and L. Sanchez-Garrido. Project statistics and programming, January 2004.
- [8] V. Airault. Application des algorithmes génétiques au jeu d’hex. Technical report, Université Paris 5, 2001. Rapport de stage de MIAIF2.

- [9] L.V. Allis. *Searching for solutions in games and artificial intelligence*. PhD thesis, Vrije Universitat Amsterdam, 1994.
- [10] L.V. Allis, M. van der Meulen, and H.J. van den Herik. Proof-number search. *Artificial Intelligence*, 66:91–124, 1994.
- [11] T. Anantharaman, M. Campbell, and F. Hsu. Singular extensions: adding selectivity to brute force searching. *Artificial Intelligence*, 43(1):99–109, 1989.
- [12] V. Anshelevich. Hexy. <http://home.earthlink.net/~vanshel/>.
- [13] V. Anshelevich. A hierarchical approach to computer hex. *Artificial Intelligence*, 134:101–120, 2002.
- [14] C. Mary B. Bouzy. Etat actuel de la programmation du jeu de go. *Revue Flux*, (184):38–42, Juin 1997. revue de l'amicale des ingénieurs de l'école supérieure d'électricité.
- [15] E. Baum and W. Smith. A bayesian approach to relevance in game-playing. *Artificial Intelligence*, 97:195–242, 1997.
- [16] D. Beal. A generalised quiescence search algorithm. *Artificial Intelligence*, 43:85–98, 1990.
- [17] E. Berlekamp. Introductory overview of mathematical go endgames. In *Proceedings of Symposia in Applied MATheatics*, volume 43, 1991.
- [18] E. Berlekamp. *Games of No Chance*, chapter The Angel problem, pages 3–12. MSRI Publications, 1996.
- [19] E. Berlekamp. *More Games of No Chance*, chapter The 4G4G4G4G4 problems and solutions, pages 231–242. MSRI Publications, 2002.
- [20] E. Berlekamp, J. Conway, and R. Guy. *Winning Ways*. Academic Press, Londres, 1982.
- [21] E. Berlekamp and D. Wolfe. *mathematical go: chilling gets the last point*. AK Peters, 1994.
- [22] H. Berliner. The b* tree search algorithm: a best-first proof procedure. *Artificial Intelligence*, 12:23–40, 1979.
- [23] H. Bernery, B. Bonnenfant, B. Jezequel, S. Saidi, and S. Thiébaut. Project statistics and programming, January 2004.
- [24] D. Billings, A. Davidson, J. Schaeffer, and D. Szafron. The challenge of poker. *Artificial Intelligence*, 134:201–240, 2002.
- [25] C. Bishop. *Neural networks and pattern recognition*. Oxford University Press, 1995.

- [26] M. Boon. A pattern matcher for goliath. *Computer Go*, 13:13–23, 1990.
- [27] M. Boon. Overzicht van de ontwikkeling van een spelend programma. Master’s thesis, 1992.
- [28] B. Bouzy. Explication de connaissances du joueur de go. Technical report, LAFORIA, La Motte d’Aveillans, Mars 1994. Poster au Premier colloque jeunes chercheurs en sciences cognitives.
- [29] B. Bouzy. Modélisation des groupes au jeu de go. Technical report, LAFORIA, Marseille, Septembre 1994. Poster au 2ème colloque jeunes chercheurs en Intelligence Artificielle.
- [30] B. Bouzy. The indigo program. In *2nd Game Programming Workshop in Japan*, pages 197–206, Hakone, 1995.
- [31] B. Bouzy. Les ensemble flous au jeu de go. In *Rencontres Francophones sur Logique Floue et Applications*, Paris, 1995.
- [32] B. Bouzy. *Modélisation cognitive du joueur de Go*. PhD thesis, Université Paris 6, 1995.
- [33] B. Bouzy. On meta-game architectures useful in the game of go. In *Workshop on Reflection and Meta-level architectures, IJCAI*, pages 80–85, Montréal, 1995.
- [34] B. Bouzy. Explication de connaissances non conscientes par modélisation computationnelle dans un domaine complexe. In *2ème Colloque Jeunes Chercheurs en Sciences Cognitives*, pages 276–279, Giens, 1996.
- [35] B. Bouzy. Spatial reasoning in the game of go. In *Workshop on Representations and Processes in Vision and Natural Language, ECAI*, pages 78–80, Budapest, 1996.
- [36] B. Bouzy. There are no winning moves except the last. In *6th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*, pages 197–202, Grenade, 1996.
- [37] B. Bouzy. Un modèle du jeu de go basé sur des interactions. In Joël Quinton J.P. Muller, editor, *IA distribuée et systèmes multi-agents*, pages 73–83, Port-Camargue, 1996. Hermès.
- [38] B. Bouzy. Incremental updating of objects in indigo. In *4th Game Programming Workshop in Japan*, pages 55–64, Hakone, 1997.
- [39] B. Bouzy. An interaction-based model for situated agents. In *International Conference on Multi-Agent Systems*, pages 399–400, Paris, 1998.
- [40] B. Bouzy. Complex games in practice. In *5th Game Programming Workshop in Japan*, pages 53–60, Hakone, 1999.

- [41] B. Bouzy. Alfa-beta enhancements vs domain-dependent enhancements through a small go board study. unpublished, 2001.
- [42] B. Bouzy. Go patterns generated by retrograde analysis. In *Computer Olympiad Workshop*, Maastricht, 2001.
- [43] B. Bouzy. Le role des concepts spatiaux dans la programmation du jeu de go. *Revue d'Intelligence Artificielle*, 15(2):143–172, 2001.
- [44] B. Bouzy. Indigo home page. www.math-info.univ-paris5.fr/~bouzy/INDIGO.html, 2002.
- [45] B. Bouzy. A small go board study of metric and dimensional evaluation functions. In Y. Björnsson J. Schaeffer, M. Müller, editor, *Computers and Games*, number 2883 in Lecture Notes in Computer Science, pages 376–392, Edmonton, 2002. Springer.
- [46] B. Bouzy. Associating knowledge and monte carlo approaches within a go program. In *7th Joint Conference on Information Sciences*, pages 505–508, Raleigh, 2003.
- [47] B. Bouzy. La conférence computers and games 2002. *Revue d'Intelligence Artificielle*, 17(4):687–693, 2003.
- [48] B. Bouzy. Mathematical morphology applied to computer go. *International Journal of Pattern Recognition and Artificial Intelligence*, 17(2):257–268, March 2003.
- [49] B. Bouzy. The move decision process of indigo. *International Computer Game Association Journal*, 26(1):14–27, March 2003.
- [50] B. Bouzy. Associating knowledge and monte carlo approaches within a go program. *Information Sciences*, 2004. à paraître, version améliorée de celle de JCIS.
- [51] B. Bouzy. Associating shallow and selective global tree search with monte carlo for 9x9 go. In *Proceedings of the fourth Computers and Games conference (CG04)*, 2004.
- [52] B. Bouzy and T. Cazenave. Shared concepts between complex domains and the game of go. In *International Conference on Context*, Rio de Janeiro, 1997.
- [53] B. Bouzy and T. Cazenave. Computer go: an ai oriented survey. *Artificial Intelligence*, 132:39–103, 2001.
- [54] 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.

- [55] B. Bouzy and B. Victorri. Go et intelligence artificielle. *Bulletin de l'AFIA*, (10), Juillet 1992.
- [56] M. Brooks. Go for it. *New Scientist*, pages 38–40, April 2002.
- [57] B. Brügmann. Monte carlo go. www.joy.ne.jp/welcome/igs/Go/computer/-mcgo.tex.Z, 1993.
- [58] D. Bump. Gnugo home page. www.gnu.org/software/gnugo-devel.html, 2003.
- [59] M. Buro. *Methods for the evaluation of game positions using examples*. PhD thesis, Paderborn University, 1994.
- [60] M. Buro. Probcut: an effective selective extension of the alpha-beta algorithm. *ICCA Journal*, 18(2):71–76, 1995.
- [61] M. Campbell, A.J. Hoane, and F-H Hsu. Deep blue. *Artificial Intelligence*, 134:57–83, 2002.
- [62] M. Campbell and T. Marsland. A comparison of minimax tree search algorithms. *Artificial Intelligence*, 20:347–367, 1983.
- [63] A-E. Carsin, A. Grand, M. Le Guen, C. Roy, and C. Saillé. Project statistics and programming: Amazon game and monte carlo's method, January 2004.
- [64] S. Castillo and J.F. Goudou. Programmation d'un logiciel de go. méthode de monte carlo. Technical report, Ecole Nationale Supérieure des Télécommunications, 2003. Projets.
- [65] T. Cazenave. Golois. www.ai.univ-paris8.fr/~cazenave/Golois.html.
- [66] T. Cazenave. *Système d'apprentissage par auto-observation. Application au jeu de Go*. PhD thesis, Université Paris 6, 1996.
- [67] T. Cazenave. Abstract proof search. In I. Frank T. Marsland, editor, *Computers and Games*, number 2063 in Lecture Notes in Computer Science, pages 39–54. Springer, 2000.
- [68] T. Cazenave. A generalized threats search algorithm. In Y. Björnsson J. Schaeffer, M. Müller, editor, *Computers and Games*, number 2883 in Lecture Notes in Computer Science, pages 75–87. Springer, 2002.
- [69] T. Cazenave. Gradual abstract proof search. *International Computer Game Association Journal*, 25(1):3–15, March 2002.
- [70] K. Chen. The move decision process of go intellect. *Computer Go*, 14:9–17, 1990.
- [71] K. Chen. Some practical techniques for global search in go. *ICGA Journal*, 23(2):67–74, 2000.

- [72] K. Chen. A study of decision error in selective game tree search. *Information Sciences*, 135:177–186, 2001.
- [73] K. Chen. Gnu go wins 19x19 go tournament. *International Computer Game Association Journal*, 26(4):261–262, December 2003.
- [74] K. Chen and Z. Chen. Static analysis of life and death in the game of go. *Information Sciences*, pages 113–134, 1999.
- [75] A. Cohn. Qualitative spatial representation and reasoning techniques. In *Proceedings KI 91*, volume 1303 of *lecture Notes in Artificial Intelligence*, pages 1–30. Springer, 1997.
- [76] J. Conway. *On Numbers and Games*. Academic Press, 1976.
- [77] F. Dahl. Honte, a go-playing program using neural nets. In J. Fürnkranz and M. Kubat, editors, *Workshop at the 16th International Conference on Machine Learning (ICML-99)*, Bled, Slovenia, June 1999.
- [78] C. Donninger. Null move and deep search: selective search heuristics for obtuse chess programs. *ICCA Journal*, 16(3):137–143, 1993.
- [79] M. Egenhofer. Reasoning about binary topological relations. In *Proceedings Advance in Spatial Databases*, volume 525 of *Lecture Notes in Computer Science*, pages 143–160. Springer, 1991.
- [80] M. Enzenberger. Computer go bibliography. www.markus-enzenberger.de/.
- [81] M. Enzenberger. Neurogo. www.markus-enzenberger.de/neurogo.html.
- [82] M. Enzenberger. The integration of a priori knowledge into a go playing program. www.markus-enzenberger.de/, 1996.
- [83] M. Enzenberger. Evaluation in go by a neural network using soft segmentation. In Ernst A. Heinz H. Jaap van den Herik, Hiroyuki Iida, editor, *10th Advances in Computer Games*, pages 97–108, Graz, 2003. Kluwer Academic Publishers.
- [84] D. Fotland. The many faces of go. www.smart-games.com/manyfaces.html.
- [85] D. Fotland. An algorithm for simple ladders. *Computer Go*, 2:5–7, 1987.
- [86] D. Fotland. Static eye in "the many faces of go". *ICGA Journal vol 25 n4*, pages 203–210, December 2002.
- [87] A. Fraenkel. *Games of No chance*, volume 29, chapter What is a game?, pages 43–60. MSRI Publications, 1996.

- [88] R.D. Greenblatt, D.E. Eastlake, and S.D. Crocker. The greenblatt chess program. In *Fall Joint Computing Conference*, volume 31, pages 801–810, New York ACM, 1967.
- [89] A. Junghanns. Are there practical alternatives to alpha-beta? *ICCA Journal*, 21(1):14–32, March 1998.
- [90] A. Kierulf. Smartgo. www.smartgo.com/.
- [91] A. Kierulf, K. Chen, and J. Nievergelt. Smart game board and go explorer: a study in software and knowledge engineering. *Communications of the ACM*, 33(2):152–166, February 1990.
- [92] D. Knuth and R. Moore. An analysis of alpha-beta pruning. *Artificial Intelligence*, 6:293–326, 1975.
- [93] R. Korf. Depth-first iterative deepening: an optimal admissible tree search. *Artificial Intelligence*, 27:97–109, 1985.
- [94] R. Korf and D. Chickering. Best-first search. *Artificial Intelligence*, 84:299–337, 1994.
- [95] B. Kuipers. Modeling spatial knowledge. *Cognitive Science*, 2:129–153, 1978.
- [96] H. Landman. *Games of No chance*, volume 29, chapter Eyespace Values in Go, pages 227–258. MSRI Publications, 1996.
- [97] D. Lichtenstein and M. Sipser. Go is polynomial space hard. *Journal of the ACM*, 2:393–401, 1980.
- [98] J. Lieberum. Amazong. web2.whosting.ch:1412/jenslieb/amazong/amazong.html.
- [99] J. Lieberum. An evaluation function in the game of amazons. In Ernst A. Heinz H. Jaap van den Herik, Hiroyuki Iida, editor, *10th Advances in Computer Games*, pages 299–308, Graz, 2003. Kluwer Academic Publishers.
- [100] S. Lucas and G. Kendall. Ieee symposium on computational intelligence in games. cswww.essex.ac.uk/Research/cigames/, April 2005.
- [101] K. Marignac, J. Petit, M. Ribeiro, and G. Sprauel. Project statistics and programming, January 2004.
- [102] T. Marsland. A review of game-tree pruning. *ICCA Journal*, 9(1):3–19, 1986.
- [103] G. Martin. *More Games of No Chance*, volume 42, chapter Restoring Fairness to DukeGo, pages 79–88. MSRI Publications, 2002.
- [104] D. McAllester. Conspiracy numbers for min-max search. *Artificial Intelligence*, 35:287–310, 1988.

- [105] R. Moneret. *Stratégos: un système multi-jeux utilisant la théorie combinatoire des jeux, capable d'apprendre automatiquement les dépendances entre sous-jeux locaux*. PhD thesis, Université Paris 6, 2000.
- [106] M. Müller. Explorer. web.cs.ualberta.ca/~mmueller/cgo/explorer.html.
- [107] M. Müller. Decomposition search: A combinatorial games approach to game tree search, with applications to solving go endgame. In *IJCAI*, pages 578–583, 1999.
- [108] M. Müller. Global and local game tree search. *Information Sciences*, 135:187–206, 2001.
- [109] M. Müller. Computer go. *Artificial Intelligence*, 134:145–179, 2002.
- [110] M. Müller and R. Gasser. *Games of No Chance*, volume 29, chapter Experiments in Computer Go endgames, pages 273–284. MSRI Publications, 1996.
- [111] M. Müller and T. Tegos. *More Games of No Chance*, volume 42, chapter Experiments in Computer Amazons, pages 243–260. MSRI Publications, 2002.
- [112] A.J. Palay. *Searching with probabilities*. Morgan Kaufman, 1985.
- [113] J. Pearl. Scout: a simple game-searching algorithm with proven optimal properties. pages 143–145.
- [114] J. Pearl. Asymptotic properties of minimax trees and game-searching procedures. *Artificial Intelligence*, 14:113–138, 1980.
- [115] J. Pitrat. *Métaconnaissance, futur de l'intelligence artificielle*. Hermès, 1990.
- [116] A. Plaat, J. Schaeffer, W. Pils, and A. de Bruin. Best-first fixed depth minimax algorithms. *Artificial Intelligence*, 87:255–293, November 1996.
- [117] M. Reiss. Go++. www.goplusplus.com/.
- [118] M. Reiss. Mick's computer go page. www.reiss.demon.co.uk/webgo/compgo.htm.
- [119] P. Ricaud. *Gobelin: une approche pragmatique de l'abstraction appliquée à la modélisation de la stratégie élémentaire au jeu de go*. PhD thesis, Université Paris 6, 1996.
- [120] P. Ricaud. A model of strategy for the game of go using abstraction mechanisms. In *Proceedings IJCAI*, pages 678–683, Nagoya, Japan, 1997.
- [121] R. Rivest. Game-tree searching by min-max approximation. *Artificial Intelligence*, 34(1):77–96, 1988.

- [122] A. Sadikov, I. Bratko, and I. Kononenko. Search versus knowledge: an empirical study of minimax on krk. In Ernst A. Heinz H. Jaap van den Herik, Hiroyuki Iida, editor, *10th Advances in Computer Games*, pages 33–44, Graz, 2003. Kluwer Academic Publishers.
- [123] A. Samuel. Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, 3:210–229, 1959.
- [124] A. Samuel. Some studies in machine learning using the game of checkers: Recent progress. *IBM Journal of Research and Development*, 11:601–617, 1967.
- [125] J. Schaeffer. The history heuristic and alpha-beta search enhancements in practice. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1203–1212, November 1989.
- [126] J. Schaeffer. Conspiracy numbers. *Artificial Intelligence*, 43:67–84, 1990.
- [127] J. Schaeffer. *One Jump Ahead: Challenging Human Supremacy in Checkers*. Springer-Verlag, 1997.
- [128] J. Schaeffer. The games computers (and people) play. *AAAI*, 2000.
- [129] J. Schaeffer and J. van den Herik. Games, computers, and artificial intelligence. *Artificial Intelligence*, 134:1–7, 2002.
- [130] N. Schraudolf, N. Dayan, and T. Sejnowski. Temporal difference learning of a position evaluation in the game of go. In Cowan, Tesauro, and Alspector, editors, *Neural Information Processing Systems*, volume 6, pages 817–824. Morgan Kaufmann, San Francisco, 1994.
- [131] S. Sei and T. Kawashima. A solution of go on 4x4 board by game tree search program. In *4th game Informatics Group Meeting in Japan*, pages 69–76, 2000. in Japanese.
- [132] J. Serra. *Image Analysis and Mathematical Morphology*. Academic Press, London, 1982.
- [133] C.E. Shannon. Programming a computer to play chess. *Philosoph. Magazine*, 41:256–275, 1950.
- [134] B. Sheppard. World-championship-caliber scrabble. *Artificial Intelligence*, 134:241–275, 2002.
- [135] D.J. Slate and L.R. Atkin. Chess 4.5 - the northwestern university chess program. In P. Frey, editor, *Chess Skill in Man and Machine*, pages 82–118. Springer-Verlag, 1977.
- [136] R. Snatzke. *More Games of No Chance*, volume 42, chapter Exhaustive Search in Amazons, pages 261–278. MSRI Publications, 2002.

- [137] B. Spight. *More Games of No Chance*, chapter Go Thermography: the 4/21/98 Jiang-Rui Endgame, pages 89–106. MSRI Publications, 2002.
- [138] G.C. Stockman. A minimax algorithm better than alpha-beta ? *Artificial Intelligence*, 12:179–196, 1979.
- [139] R. Sutton. Learning to predict by the method of temporal differences. *Machine Learning*, 3:9–44, 1988.
- [140] R. Sutton and A. Barto. *Reinforcement Learning: an introduction*. MIT Press, 1998.
- [141] M. Tajima and N. Sanechika. Estimating the possible omission number for groups in go by the number of n-th dame. In J. van den Herik and H. Iida, editors, *Proceedings of the First International Conference on Computers and Games*, volume 1558 of *Lecture Notes in Computer Science*, pages 265–281. Springer, 1998.
- [142] T. Tegos. The game of amazons. web.cs.ualberta.ca/~tegos/amazons/.
- [143] G. Tesauro. Temporal difference learning and td-gammon. *Communications of the ACM*, 38:58–68, 1995.
- [144] G. Tesauro. Programming backgammon using self-teaching neural nets. *Artificial Intelligence*, 134:181–199, 2002.
- [145] 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.
- [146] T. Thomsen. Lambda-search in game trees, with application to go. *International Computer Game Association Journal*, 23(4):203–217, December 2000.
- [147] K. Thomson. Retrograd analysis of certain endgames. *ICCA Journal*, 9(3):131–139, 1986.
- [148] K. Thomson. 6-piece endgames. *ICCA Journal*, 19:215–226, 1996.
- [149] E. Thorpe and W. Walden. A partial analysis of go. *Computer Journal*, 7(3):203–207, 1964.
- [150] E. Thorpe and W. Walden. A computer assisted study of go on mxn boards. *Information Sciences*, 4:1–33, 1972.
- [151] Erika Valencia, Gérard Ligozat, and Jean-Paul Sansonnet. Séminaire quando et urbi. www.limsi.fr/Individu/erika/QetU.html.
- [152] 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.

- [153] E. van der Werf. Aya wins 9x9 go tournament. *International Computer Game Association Journal*, 26(4):263, December 2003.
- [154] E. van der Werf, H.J. van den Herik, and J.W.H.M. Uiterwijk. Solving go on small boards. *International Computer Game Association Journal*, 26(2):92–107, June 2003.
- [155] J. van Rijswijck. Computer hex: Are bees better than fruitflies? Master’s thesis, Alberta University, 2000.
- [156] J. von Neumann and O. Morgenstern. *Theory of Games and Economic behavior*. Princeton University Press, 1944.
- [157] C. Watkins. *Learning from delayed rewards*. PhD thesis, Cambridge University, 1989.
- [158] C. Watkins and P. Dayan. Q-learning. *Machine Learning*, 8:279–292, 1992.
- [159] N. Wedd. Goemate wins go tournament. *International Computer Game Association Journal*, 23(3):175–177, September 2000.
- [160] P. Woitke. Goahead. astro.physik.tu-berlin.de/~woitke/GoAhead.html.
- [161] T. Wolf. The program gotools and its computer-generated tsume-go database. In *1st Game Programming Workshop in Japan*, pages 84–91, 1994.
- [162] T. Wolf. About problems in generalizing a tsumego program to open positions. In *3rd Game Programming Workshop in Japan*, pages 20–26, 1996.
- [163] T. Wolf. Forward pruning and other heuristic search techniques in tsume go. *Information Sciences*, 122:59–76, 2000.
- [164] H. Yamashita. Hiroshi’s computer shogi and go. www32.ocn.ne.jp/~yss/index.html.
- [165] A. Zobrist. A model of visual organization for the game of go. In *Proceedings AFIPS Spring Joint Computer Conference*, pages 103–111, Boston, 1969. AFIP Press.
- [166] A. Zobrist. A new hashing method with application for game playing. *ICCA Journal*, 13(2):69–73, 1990.