ゲーム情報学 5－2
（2001. 3. 16）

The Butterfly and the Board:
A preliminary look at chaos theory as a model for robust Go algorithms

Uri Globus *globus@cs.inf.shizuoka.ac.jp*
Department of Computer Science, Shizuoka National University, Japan

Markian Hlynka *markian@cs.ualberta.ca*
Department of Computing Science, University of Alberta, Canada

The game of Go is extraordinarily difficult for computers. Its scope far outstrips conventional game-programming techniques [1]. This is reflected in the fact that Go is identified as one of the "Challenge Problems" of AI [4]. Go's search space is simply too large to be tackled in its entirety. The result, an unavoidable lack of robustness in many current approaches, indicates the difficulties with conventional models. Thus, new models are required to render Go in its entirety a feasible undertaking. Preliminary work toward this end led to the consideration of nonlinear systems and chaos theory.

Keywords: computer Go, artificial intelligence, robustness, chaos theory

1.0 Introduction

Go, an oriental game several times older than western chess, is extraordinarily difficult for computers. Based upon simple rules, the scope of the game and possible interactions far outstrip conventional game-programming techniques [1]. Go is identified as one of the "Challenge Problems" of AI [4]. Go programs remain relatively poor performers when compared to humans, and are certainly not up to the par of other game programs that play Chess, Checkers, Othello, and other traditional games. Go programs exhibit inconsistent play characterized by good to excellent moves with poor follow-up play. Their weaknesses are readily apparent to human players. Even with incredibly large handicaps, master players have documented wins against some of the best programs [2].

Martin Müller [2] identifies several key challenge areas for Computer Go. These are: Developing a program whose strength increases with increased computational resources; Developing a program that understands and correctly uses threats and forcing moves; Developing a program that will consistently win handicap games; Integrating the existing expert modules into a program able to combine them effectively; Solving Go on small boards; Developing a test suite of Go problems; Developing a public source code library of common functions [2].

A common thread to the above is that current programs demonstrate a lack of robustness. It is the goal of this paper to present some definitions and ideas, and to link them together with this common thread. In the course of this discussion, a potential for computer Go discourse and experimentation centered upon a chaos theoretic approach is identified.

1.1 State of the Art

M. Müller states in [2] that current Go programs can play a "respectable" game of Go. However, they can also lose games even when given ridiculous handicaps [2]. Analysis of games and literature alike reveals that while computers playing other computers can appear to play well, this is because a computer is unable to exploit weaknesses to which it is itself susceptible. To a human, these weaknesses are glaringly obvious, and result in losses even when the computer has a huge handicap [2]. More robust algorithms, while not necessarily eliminating these weaknesses, would play to minimize their impact.

2.0 Some Definitions

Several definitions are necessary to add fluidity and lucidity to the following discussions. These involve primarily, though not exclusively, robustness, search spaces, chaos and fractals.

2.1 Robustness

In the context of this paper, robustness has two meanings. First, it means that an algorithm performs consistently. If a program plays at a 5 kyu level in some games, and at a 20 kyu level ten games later, clearly the program cannot claim a strength of 5 kyu. Certainly, it can be argued that against a human player, the human has the advantage of adaptability. However, it is rather rare for a human to improve several levels in the course of ten games. Rather, the human has learned to exploit a weakness in her opponent. If the opponent has such a weakness, it is clearly not as strong as it first appeared.

Second, a robust algorithm in the purely computer science sense is one which is scalable in terms of computational resources. A program playing at a 20 kyu level should, with twice the processing power available, not merely play at 20 kyu twice as fast. It should play stronger as well. A one to one correspondence would be ideal but, as games such as chess and checkers have shown us, this is usually unrealistic.

Robust algorithms are essential for computer go, both for play against human opponents who are able to find and exploit weaknesses, and to effectively scale as computer hardware improves.

2.2 Search Space

When search space is discussed in artificial intelligence and games, it refers to the sum of the number of possible states at each point in an environment. More simply, it is the set of all possible states of the environment. However, for the purposes of this discussion of go programming, we would like to make a further distinction. Thus, the search space for a game will be comprised of two distinct concepts: board space and game space.

2.3 Board Space

Board space shall be used to synonymously refer to what we previous called search space. That is, the set of all possible states obtainable on the board for a particular game. We will not concern ourselves for the moment as to whether these states are necessarily legal positions.

2.4 Game Space

Game space is the set of positions which are relevant for a particular game. It is a subset of board space. Within games, it has long been known that human master players have the uncanny ability to only analyze the 'correct' lines of play. Thus, while the number of possible states which *can* be played out on the board are enormous, the number of reasonably plausible states diminishes in inverse proportion to the strength of the player.

2.5 Chaos and Fractals

Chaos theory and fractals involve systems which evince nonlinear dynamics. Nonlinear dynamics refer to systems in which small changes can result in large, unpredictable, systematic change to the whole system [6]. This is often referred to as the "butterfly effect" [7]. In the game of go, subtle changes in the placement of stones and the effects of seemingly small local skirmishes can have incredibly large and varied effects upon the final outcome of the game. Another key feature of fractal systems is self-similarity: the system looks much the same on all scales. Go is quite arguably self-similar.

3.0 Game Techniques: Background

To date, most successful championship game programs rely on some sort of heuristic search [5]. IBM's Deep Blue chess computer is essentially a marriage of computer hardware and software designed specifically to search chess positions at a blindingly fast rate. Chinook, the world man-machine checkers champion, is another example of search engineering, coupled with powerful techniques to direct and reduce that search. While both these and other such programs play on a level with, or above, human champions, no one would argue that programs emulate the thought processes of those human champions.

Naturally, there are exceptions. TDGammon and Logistello use neural network evaluations and, until recently, TDGammon did not use search to any great extent. Recently, TDGammon has been enhanced to further leverage search. While this strengthens it still further, the search in TDGammon is secondary. However, it seems certain that whatever technique is used, there is a good chance it can be strengthened with proper application of search.

However, while alpha-beta and associated algorithms, the mainstay of traditional game programs, are certainly computationally robust, they cannot compete with humans in some areas. The search spaces of these areas are simply too big, and so far no known general heuristics exist for sufficiently narrowing that search space [5]. Indeed, much of the research done in games involves extensions and enhancements directed to eliminating or reducing unnecessary compu-

tation in basic search. However, even these techniques coupled with ever-increasing hardware speeds have proved insufficient to games like Go. Thus, progress requires the development of new techniques and refinement of current ones. These techniques may well stand on their own, as in the case of neural networks mentioned above, or may further enhance current search methods.

In order to make search-based approaches viable for Go, a model is necessary which will allow the search space to be effectively (and efficiently) reduced into what we have dubbed 'game-space'. Without this model, tackling Go becomes a task ranging from difficult to impossible. (This does not discount the possibility of other methods which might not require search.) This type of approach goes back to the work of Stilman [3] in chess and military strategy. More recently Thomsen [8] has applied a similar approach to ladders in Go.

4.0 Motivation

When the authors of this paper first approached computer Go research, it was evident that many skilled people were already working on computer Go projects. Thus, it was felt that in order to do something worthwhile it would be necessary to develop a new model. New models hopefully produce better methods of evaluation, in turn leading to stronger programs. In this manner it was hoped that needless duplication of others' work would be avoided, while contributing something worthwhile to the computer Go research scene.

Initially, robustness of computer Go algorithms was identified as a potential avenue of research. Current Go programs seemed to lack robustness: their playing strengths were inconsistent nor were the algorithms computationally scalable. Therefore, a subset problem and goal were chosen: Develop algorithms for robust fuseki (opening phase).

After detailed analysis, it seemed that the first considerations in robust fuseki were group connectivity, eye space, and territory. The common thread to all of these was life and death: if you cannot reasonably determine the life and death status of groups and potential groups, concepts like territory, connectivity, and eye space become meaningless. Thus, it was set out to develop an evaluation model which encapsulated life and death for the early game stages: this means a *constructive* as opposed to *analytical* evaluation of living groups. That is, not evaluating whether a group is currently alive, but rather where to play to maximize a number of stones' chances for living. This is different from traditional life and death problems. Rather than dealing with specific life and death situations, these models were intended to deal with the *potential* life and death situations which arise in the early stages of Go.

In Go terminology this idea of constructing shapes with good potential for surviving tactical encounters is called sabaki. The idea of robust shape is implied in the Go concepts of katachi (good shape), thickness, and lightness. Thickness and lightness represent distinct strategies as regards robustness of shapes. The former is characterized by firmness; the shape is not easily attacked. The latter maintains flexibility in order to minimize losses under severe attack. These concepts play an important role in the game due to the difficulty in determining which blocks might need to survive a continuous attack. A continuous attack is where the opponent, by means of other threats, is able to attack a particular area more frequently than the defender is able to

respond. Ko fights present such double threats, which many times involve distinct parts of the board.

Several models were considered for developing algorithms which could generate good shapes. These were: Rule-sets, neural networks, and artificial life. All of these were attempts to specify with different mathematical models (logic, function approximation and Turing computability, respectively) the attributes of good and bad shape. The goal, to reiterate, was to find more robust representations. Thus, the models had to include a confidence measure. If confidence was less than one hundred percent, the model should take appropriate steps to minimize damage. (Take the less-dangerous and more confident alternative.) Certain moves are robust, for example, because they don't lead to tactical complexities. In an uncertain situation one's own territory should be secured before more risky endeavors are attempted.

The problem encountered was twofold. First, it was difficult to scale algorithms to different sizes of playing area: when should a move encompass the entire board and when should it be more local? Second, how to distinguish good versus bad shapes mathematically, without endless pattern databases? The stumbling stone to both these facets was how to effectively set temporary boundaries for evolving shapes on the board.

A potential solution presented itself: During the repeated analysis of several similar but distinctly good and bad shapes on the board, it was remarked that the good shapes appeared to possess a higher fractal dimension. From there it was also suggested that Go exhibits self-similarities and boundary conditions which are also encountered in nonlinear dynamics. Finally, it is known that chaotic systems are sometimes kept 'stable' through the application of a continuous series of perturbations [11]. Thus, the idea of a chaos theory of Go was brought forth.

5.0 Chaos and Go

One of the most important features of chaotic structures is self-similarity [6]. That is, the fact that on different scales the system exhibits identical features. Many traditional games, such as chess, are not fundamentally self-similar due to constraints which the rules place upon the game. A chess position is meaningless without two kings on the board. Thus, a chess game can not necessarily be decomposed into sub-games which still possess the characteristics of the entire game.

We believe that Go is different. A Go position can potentially be decomposed into self-similar sections; each sub-game has all the characteristics of the whole game. Granted, the combination of these games is a non-trivial task (as is the actual decomposition). However, the actual mechanics of a decomposition are not necessarily requisite. Rather, the plausibility of such a decomposition leads to the suggestion that other fractal structures might be reasonably expected to produce interesting results when applied to computer go.

Go also exhibits self-similarity on more subtle levels than its physical structure. Important strategic concepts of Go [10] are also self-similar. Human players intuitively apply these concepts simultaneously to the large scale of the entire game, to the small scale of tactical skirmishes, and to every level in between. While computers have proven effective in applying these strategies to

small scale tactics, they often lack the ability to apply these concepts to higher levels of abstraction.

Since chaotic systems, like Go, are highly sensitive to varying initial conditions, it is hypothesized that nonlinear dynamics might provide the necessary tools to determine new measures of the influence of stones and their stability. Chaos dynamics seem to offer a way to continuously perturb systems into preferred states or, at least, less undesirable ones. Thus, chaos should be explored as a way to provide methods to suggest moves which are likely candidates for perturbing the game state in a particular direction. If this effect is pronounced enough, such methods could be incorporated into much more effective searches: searches which could be focused upon the *game space*.

6.0 Directions for Future Work

Research is now in progress to fully understand the nature of chaotic systems and which elements are best suited for application to computer Go.

7.0 References

1. M. Müller. Not like other games - why tree search in Go is different. In Proceedings of Fifth Joint Conference on Information Sciences (JCIS 2000), pages 974-977, 2000.
2. M. Müller. Computer Go, 2000. Survey paper, special issue on games of Artificial Intelligence Journal, to appear.
3. B. Stilman. Linguistic geometry for symmetric and asymmetric war games. In Proceedings of Second International Conference on Computers and Games (CG 2000), 2000.
4. B. Selman, R. Brooks, T. Dean, E. Horvits, T. Mitchell & N. Nilsson. Challenge Problems for Artificial Intelligence. AAAI-96, Thirteenth National Conference on Artificial Intelligence, AAAI Press, August 1996.
5. J. Burmeister & Janet Wiles. An Introduction to the Computer Go Field and Associated Internet Resources. On-line reference: http://www2.psy.uq.edu.au/~jay/go/CS-TR-339.html
6. Academic Press, Inc. chaos, Definition from AP Dictionary of Science and Technology. On-line reference: http://www.academicpress.com/inscight/08261998/chaos2.htm
7. The Columbia Electronic Encyclopedia, Nonlinear Dynamics. On-line reference: http://kids.infoplease.lycos.com/ce6/sci/A0835836.html Columbia University Press, 2000.
8. T. Thomsen. Lambda-search in game trees - with application to go. CG 2000 (see [4])
9. Robert A. Levinson. Exploiting the physics of state-space search. Technical Report UCSC-CRL-94-32, Department of Computer and Information Sciences, University of California, Santa Cruz, CA, 1994.
10. Y. Nagahara. Strategic concepts of Go. The Ishi Press, Inc. Tokyo, Japan. 1972.
11. E. Weeks & J. Burgess. Controlling chaos with neural networks. On-line reference: http://chaos.ph.utexas.edu/research/dsane/slog.html