A Dynamical Systems Approach for Static Evaluation in Go

Thomas Wolf
Brock University
St. Catharines, Ontario, Canada
WWW: http://www.brocku.ca/mathematics/people/wolf/
Email: twolf@brocku.ca

***Abstract*—In the paper arguments are given why the concept of static evaluation has the potential to be a useful extension to Monte Carlo tree search. A new concept of modeling static evaluation through a dynamical system is introduced and strengths and weaknesses are discussed. The general suitability of this approach is demonstrated.**

I. Motivation

The concept of Monte-Carlo simulations applied to Go [1] combined with the UCT algorithm [2], [3], which is a tree search method based on Upper Confidence Bounds (UCB) (see e.g. [4]) produced a new type of programs like [5], [6], [7] that dominate computer Go in recent years. The detailed tournament report [8] of the program MoGo playing against professional and amateur players reveals strengths and weaknesses of MoGo which are typical for programs that perform a Monte Carlo tree search (MCTS).

Programs performing MCTS can utilize ever increasing computing power but in their pure form without extra Go knowledge the ratio log(increase in needed computing power) / (increase in strength) is too big to get to professional strength on large boards in the foreseeable future. Therefore in recent years Go knowledge has been incorporated either in form of heuristics, or pattern databases learned from professional games or from self-play. Although treesearch was naturally slowed down the playing strength increased further.

With all of this tremendous progress of MCTS compared to the knowledge based era of computer Go summarized in [9], [10], [11], it needs good reasons to start work on a static evaluation function (SE) in Go.

One indicator that more Go knowledge needs to be added is that, compared with human playing strength the playing level of current programs decreases as board size increases from 9×9 to 13×13 and then to 19×19.

The principal difficulties of deriving knowledge and applying it become more relevant as knowledge is increasingly used in MCTS.

- Knowledge that is not 100% accurate reduces the scalability of the program when enough computing power is available for global search to replace increasingly the approximate Go knowledge which then becomes less useful or even less accurate than knowledge coming from search.
- It is difficult to combine knowledge on a high level if it comes from different sources, like from pattern and from local searches. It is one of the reasons of the originally surprising success of pure MCTS that it only uses knowledge from one source (statistics of simulations) without the need of merging different types of knowledge.
- Go knowledge is either accurate, for example, when coming from life and death computations of fully enclosed regions or from matching large patterns but then these pure situations occur only rarely, or, Go knowledge is produced more often, for example, by simply counting liberties or matching small pattern or by doing quick life and death computations in open regions but then the knowledge has little significance or is not very accurate.

What is needed is an approach that has an understanding of the local and global situation in *general* positions, that either is accurate to some extent or indicates in which areas it is not accurate. On top of that it needs to be fast.

The SE proposed in this paper is applicable to any position, it is reasonably fast and has already in its first and simplest version as studied in this paper a good understanding of the situation. It is expandable to include linking and life and death information in future without having to combine knowledge in a simplistic way, for example, by taking an average with constant weights.

In section II a close look at characteristic features of the game Go provides arguments for using a dynamical systems approach as the starting point for a static evaluation function. Section III gives an introduction to discrete dynamical systems and a description of their potential benefits. The algorithmic details of formulating, initializing and evaluating dynamical systems are described in section IV. Section V comments on limitations of any dynamical systems approach that are based on its static nature. Section VI discusses the robustness of the approach (existence, uniqueness and stability of fixed points) and its efficiency in predicting moves in professional games. A comparison with the efficiency of other approaches is made in section VII. This section also contains a detailed characterization of static evaluation through dynamical systems. Section VIII includes comments on necessary extensions, especially the merger with MCTS, but also comments on how an appropriate influence function can be used in a game. The paper concludes with a short summary in section IX and an appendix that gives an example how an influence function can guide finding the best move even in situations when a sacrifice move is necessary.

II. Initial Considerations

A. Resources Unused in Pure MCTS

Why should it be possible to design a static evaluation which can provide at least some information faster and/or better than MCTS?

One strength of MCTS is to be useful for programs playing other games than Go or even for work on tasks not involving games. This strength is at the same time a weakness when applied to Go: MCTS does not take advantage of simplifying aspects of the nature of Go:

1) *Blocks are local.* Blocks connect adjacent stones of the same colour into a unit, so that either all of them are captured or none of them is. Stones that are not connected to the block do not belong to it.
2) *Capturing a block is local.* To capture a block the opponent must fill all of the block's *adjacent* intersections. Opposing stones placed further away do not capture the block.
3) *Go has an influence field.* At all stages of a game except its very end it is useful to introduce a field of 'influence' or 'strength' that guides the player (humans and computer) towards optimal play.
   This influence field is not simply a tool to accommodate human slowness in reading compared to MCTS. The point to make is that this influence field is in some sense real and can be characterized and modeled. It shows, for example, some stability or null-sum property and can explain higher playing level sacrifice moves as done with a position in the appendix.
   We want to state it as a conjecture: *To maximize playing strength for a given amount of computational power (size of memory and cycles per second, both sufficiently large but fixed) a field embodying strength or influence and perhaps other fields or state variables have to be introduced.* This is not different from progress in the natural sciences and mathematics where the improvement of quantitative knowledge and an evolution of the scientific language depend on each other.
   Knowledge on strength and influence is crucial for human players but also increasingly used in MCTS. Deriving knowledge purely from search as in original MCTS results in a search space growing exponentially with the area of the board. In contrast, the cost of computing a good static evaluation does not increase exponentially with the board size.
4) *Influence varies smoothly.* The influence of stones falls off smoothly, at least in the opening in non-life-and-death situations. Also, the example in the appendix shows the need for intermediate influence values other than 1 (full domination) and 0 (no influence at all). More discussion on smoothness in Go can be found in section II.A. in [12].

B. Design Decisions

The above observations lead to the following design decisions for the static evaluation function.

- We want to start by modeling/computing only a minimal number of variables describing a board position. This will be for each point (*empty* intersection) a number indicating whether it is under Black or White influence and for each block a measure of its strength, i.e. a probability of not being captured.
- The strength values of blocks and influence values at points are represented by floating point numbers because of points 3) and 4) above but also to evaluate some fuzzy knowledge by a number that changes smoothly with the degree of certainty of the knowledge.
- Because of points 1) and 2) above, the set of all relations of neighbouring points and blocks is formulated as a single discrete dynamical system of algebraic relations expressing the strength of each block and influence at a point in terms of the strength of neighbouring blocks and influence at neighbouring points. A more detailed description of dynamical systems is given in the subsection below. In the approach to be described a strength value is assigned to a whole block, irrespective of its size or shape. All that matters in this approximation are the neighbourhood relations. A **S**tatic **E**valuation based on **D**ynamical **S**ystems will be abbreviated as SEDS in the remainder of the paper.

A different question is whether the SE should depend on who moves next. Although it may become slightly better by taking that into account (e.g. if two important blocks of opposite colour touch each other and have only one liberty each, so that the side to move next may capture the opponent's block), the SE to be described in this contribution does not use who moves next. The reasons for this are:

- It is not obvious how to use who moves next without prejudice even in the simple case of, say, Black moving next and white blocks under atari being so small that their capture has low priority. Another example is the case when many white blocks are under atari.
- Making the SE dependent on who moves next is not a general solution. It may take 2 moves to simplify the all-or-nothing fight so that the SE can 'see' the outcome, or 3, 4, ... moves. The issue of merging SE and MCTS has to be solved more rigorously, not by a quick fix of making SE dependent on who moves next.
- The value of moving next naturally varies from area to area. To consider it properly would imply to know the value for each area but that essentially means to be able to play perfectly. This would be contradictory to the philosophy of splitting up the problem of determining the best move into three parts: designing a static evaluation, a search procedure (MCTS) and an interplay between both.

III. About Discrete Dynamical Systems

A. General Comments

A discrete dynamical system (abbreviated as dynamical system (DS) in the following) is a set of n relations between n variables where each relation takes the form of expressing one variable in terms of all others:

$$v_i = f_i(v_j) \quad (1)$$

where f is some (not necessarily continuous) map.[1]

Given numerical initial values $v_i = v_{i0}$ the system of relations can be used to compute new values $v_{i1}, v_{i2}, ..$ until after k iterations a fixed point is reached, i.e. v_i change only insignificantly:

$$|v_{ik} - v_{i(k-1)}| \leq \varepsilon \quad \forall i$$

for some threshold parameter ε, or, until k reached an upper bound $k_{\max}$.

In our approach the state variables v_i are strength values of blocks and influence values at points (empty intersections). By choosing suitable maps f_i the iteration of the system (1) should model, for example, the fact that blocks are weakened when being under strong opponent influence and as a consequence in the next iteration will become less influential on their surrounding.

A dynamical system is called *sparse* if the f_i involve only few variables. The DSs we aim at are sparse due to locality properties 1), 2) above: each empty point has at most 4 neighbours (points or blocks) and each block has only a small subset of all points and blocks as direct neighbours. If a DS is sparse then iterations are computed faster than if it would be *dense*, i.e. if many f_i would involve many v_j.

Although only local relations are recorded in the formulation of a DS, the fixed points that are computed are global properties of the complete DS due to the iterations that take place.

B. Potential Benefits of Dynamical Systems

The potential benefits of using DS to describe a board position in Go are manifold.

- It allows a consistent formalism where the influence originating from a block depends on the strength of the block and the strength of the block depends on the degree of ownership it has on neighbouring empty points.
  In other computer Go programs which use influence functions or pattern matching either all blocks have the same strength or are distinguished only between being alive or dead. The error made with this crude simplification needs to be compensated by global search, i.e. by giving the influence function and pattern heuristic relatively little weight compared to the weight of search. Whether supposedly better influence and strength values computed from finding fixed points of a DS are worth the extra effort remains to be shown but the potential is at least there.
- Data about which points and blocks are neighbour to each other are recorded to speed up the computation of f_i in (1). These data are available for other computations, like life and death.
- The strength value of blocks and influence values at points become available as a side product which may be useful for separate tactical investigations. Especially pattern matching is very popular among computer Go programmers since the very start of computer Go. Having adequate strength and influence values may allow more refined pattern which encapsulate not just local data but global board properties because the numerical strength and influence values used in the local pattern are properties of the computed fixed point, i.e. their values take into consideration the complete board. As another example, strength values of blocks may be useful to initialize the area of local life & death computations.
- The number of iterations needed for $|\Delta v_i|$ to fall below ε turns out to be a good measure for the stability of a local region on the board and thus a strong indicator for which moves need to be searched, e.g. through (some local version of) MCTS.

C. Computational Complexity

For a programmer of MCTS the computational effort to formulate a DS and find a fixed point may be horrifying. Afterall, in MCTS tens of thousands of whole games starting from a position are played just to find the next move. Speed in performing moves is essential in this approach, even to a point that it may pay off to allow illegal moves in order to save time and play more simulated games.

In total contrast a dynamical system has several hundred variables and as many equations (489 variables and relations for the full board position in diagram 2 further below). At first the DS has to be established in some form, i.e. enough data need to be collected and stored to compute the f_i in (1) efficiently. Then this system has to be iterated repeatedly to compute a fixed point. Next, an estimated score is to be computed by summing over all blocks their product (strength $\times$ size) and summing over the influence values of all empty points. All of this computation has to be done after performing each one of the legal moves that are to be evaluated to select the move with the highest score.

But, *if* local relations 1), 2) are an important part of the game, and *if* the strength of blocks and the influence at empty points are inherent characteristics of a board position then a DS (1) is the most direct and adequate formulation of all relationships on the board and solving such a systems (finding the fixed points) is the most efficient way to characterize the board position. Other methods will have at least as high costs whether obvious or hidden.

These are a lot of *if*'s which are not always satisfied. For example, in the case of a lengthy winding ladder there is no alternative to performing the tree search and computing

[1] If v_i would be functions of a parameter, e.g. time, then dynamical systems typically express the time derivative of each variable in terms of all variables (not their time derivatives).

the ladder. But many parts of the board especially in the opening are calm where the *if*'s are satisfied and where the formulation and solution of a DS appears to be the appropriate computation.

Time measurements shown at the end of section VI and comparisons with other evaluation functions in section VII indeed show that the DS-approach is fast.

Another useful feature is that the evaluation of different moves based on their score according to SEDS can be done in parallel. Especially when more life & death investigation will be added to SEDS to make it more accurate, the parallelization will become more coarse grain and thus be more effective.

IV. A Dynamical System Representing a Board Position

In this section we describe a conceptually simple dynamical system model that was implemented and studied for its strengths and weaknesses.

A. The Setup

The elementary objects on the board (we call them units from now on) are taken to be all points (empty intersections) and blocks (for which no shape is recorded). Individual stones of a block have no own identity in this model.

Based on the capture rule of Go, units have completely local relations with each other, i.e. the state variables describing each unit can be computed explicitly from the state variables of neighbouring units and the resulting dynamical system can be solved iteratively.

This system couples all units on the board (i.e. all (empty) points and blocks) and thus a fixed point of the dynamical system is a global consequence of the whole board. A change in strength of one block would influence the strength of weak neighbouring blocks and so on but the change of influence would stop at strong blocks.

With points on the edge of the board having only 3 neighbours and in the corners only having 2 neighbours, the influence of the edge should come out properly without the need of extra artificial adjustments.

B. State Variables

To each *point* i (i.e. each empty intersection, i.e. i takes at most 19×19 different values) are attached two real floating point type numbers describing probabilities at the end of the game:

$w_i \ldots$ to be owned by ○ , i.e. to be occupied by ○ or to be a point in an alive white eye
$b_i \ldots$ to be occupied by ● , i.e. to be occupied by ● or to be a point in an alive black eye

and to each *block* j is attached one number:

$s_j \ldots$ probability for this block to survive.

All values are in the interval $0 \ldots 1$.
For explanation purposes we also introduce

$\bar{w}_i,\ \bar{b}_i \ldots$ probability that at least one neighbouring intersection of point i is occupied by resp. ○ or ● at the end of the game.

C. The Relations

If b_i, w_i are the probabilities defined above then under normal circumstances they add up to 1:

$$b_i + w_i = 1. \tag{2}$$

The only exception is a seki, for example, when point i is one of the shared liberties that is not accessible to either one side. In that case we should have $b_i = w_i = 0$ but our model will give $b_i = w_i = 0.5$ which does not change the contribution of points to the score but it may have an effect on the computed strengths s_j of the blocks in seki. The SEDS could conclude from $w_i = b_i = 0$ that both blocks are safe, but not from $w_i = b_i = 0.5$. The bottom line is that the concept of seki like that of life & death, is a discrete concept resulting from the rule that both sides alternate moves. These concepts are not properly covered in this dynamical systems approach but have to be detected separately, see the discussion in section V.

Apart from relation (2) the only other assumption we make is $\quad w_i/b_i = \bar{w}_i/\bar{b}_i\ ,\quad$ i.e.

$$w_i\bar{b}_i = b_i\bar{w}_i. \tag{3}$$

At least in the extreme cases $(\bar{w}_i, \bar{b}_i) = (1,1),\ \ (1,0),\ \ (0,1)$ this relation is correct. Also for other cases it should be suitable based on the following argument.

Under Chinese rules the aim is to occupy as many points as possible at the end of the game. A single stone can not be alive on its own if surrounded only by alive opponent stones, it has to be part of a block that is alive. Thus, the probability b_i of a point being owned by Black should increase with the probability that it has an alive black stone as neighbour, i.e. $b_i \propto \bar{b}_i$ and similarly $w_i \propto \bar{w}_i$. Relation (3) is a simple example with such a dependence.

From (2), (3) we get

$$w_i = \frac{\bar{w}_i}{\bar{b}_i}(1 - w_i) = \frac{\bar{w}_i}{\bar{b}_i} - \frac{\bar{w}_i}{\bar{b}_i}w_i$$

$$w_i\left(1 + \frac{\bar{w}_i}{\bar{b}_i}\right) = \frac{\bar{w}_i}{\bar{b}_i}$$

$$w_i = \frac{\bar{w}_i}{\bar{b}_i}\left(1 + \frac{\bar{w}_i}{\bar{b}_i}\right)^{-1} = \frac{\bar{w}_i}{\bar{w}_i + \bar{b}_i} \tag{4}$$

where $\bar{w}_i, \bar{b}_i$ have to be expressed in terms of b_j, w_j, s_j from the neighbouring points and blocks. What formula (4) achieves is to express the influence in a point in terms of the status at direct neighbouring intersections.

D. An Example Computation

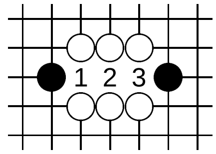

Diagram 1.

In this simple example we are going to use relation (4) to compute the probability of points 1, 2, and 3 in diagram 1 to be occupied finally by ● or ○ . For the simplicity of this example, all blocks are set to be alive: $s_j = 1,\ \forall j$. In the real model their strengths would also be computed iteratively based on the strength of direct neighbouring blocks and the influence of their direct neighbouring points.

To apply (4) we set $\bar{b}_1 = s(\text{left black stone}) = 1$ and $\bar{w}_1 = s(\text{white stones}) = 1$ and get

$$\rightarrow\ w_1 = \frac{1}{1+1} = \frac{1}{2} = b_1 = w_3 = b_3 \quad (\text{by symmetry})$$

$$\begin{array}{rcl} w_2 &=& \dfrac{1}{1+\bar{b}_2} \quad \rightarrow \bar{b}_2 \;=\; \text{probability of ● on 1 or 3} \\ && \qquad\qquad\qquad\;\; = \; 1 - \text{probability of ○ on 1 or 3} \\ &=& \dfrac{1}{1+3/4} \qquad\quad\;\; = \; 1 - w_1 w_3 \\ &=& \dfrac{4}{7} \qquad\qquad\qquad\; = \; 1 - \dfrac{1}{4} \\ b_2 &=& \dfrac{3}{7} \qquad\quad \nwarrow \qquad = \; \dfrac{3}{4} \end{array}$$

This small example demonstrates how the computation goes but it also shows the limited value of the numbers obtained. They make sense if the moves are played randomly. In the derivation of the formulae all moves are assumed to be uncorrelated, but that is not the case: if White plays on 1 then Black plays on 3 and vice-versa (if there is nothing more urgent on the board).

A similar computation is done for all blocks where the probability of a block j being captured ($= 1.0 - s_j$) is computed as the probability of all attached opponent blocks k being alive and all neighbouring points i being occupied/dominated by the opponent:

$$s_j = 1.0 - \prod_k s_k \times \prod_i \left\{ \begin{array}{ll} w_i & \text{if block } j \text{ is black} \\ b_i & \text{if block } j \text{ is white} \end{array} \right. \qquad (5)$$

Because the move taking the last liberty of a block can not be suicide, formula (5) is modified slightly. The lowest value of the w_i (resp. b_i) is increased, in the simplest choice to 1.0. In formula (5) we again, for simplicity, assume non-correlation of the feasibility of the opponent capturing moves which strictly speaking is not justified.

E. A Full Board Example

On the board in diagram 2 are 55 blocks and 217 empty points giving a system of $2 \times 217 + 55 = 489$ equations for the 217 w_i, 217 b_i and 55 s_j variables. Effectively the problem involves 217+55=271 variables because of $w_i + b_i = 1$.

The following are just three of the 489 relations:

$$\begin{array}{rcl} w_{r8} &=& (b_{q8}b_{r9}s_{r7}s_{s7} - b_{q8}b_{r9}s_{r7} + 1)/ \\ && (b_{q8}b_{r9}s_{r7}s_{s7} - b_{q8}b_{r9}s_{r7} + \\ && \;\; s_{r7}s_{s7}w_{q8}w_{r9} - s_{s7}w_{q8}w_{r9} + 2), \\ b_{r8} &=& -w_{r8} + 1, \\ s_{r7} &=& -s_{r4}s_{s7}w_{p7}w_{q6}w_{q8}w_{r8} + 1. \end{array} \qquad (6)$$

The full set is shown on `http://lie.math.brocku.ca/twolf/papers/WoSE2010/1`. Through this system each dynamical variable (2 for each point, 1 for each block) is expressed in terms of the variables describing their neighbouring points and blocks.

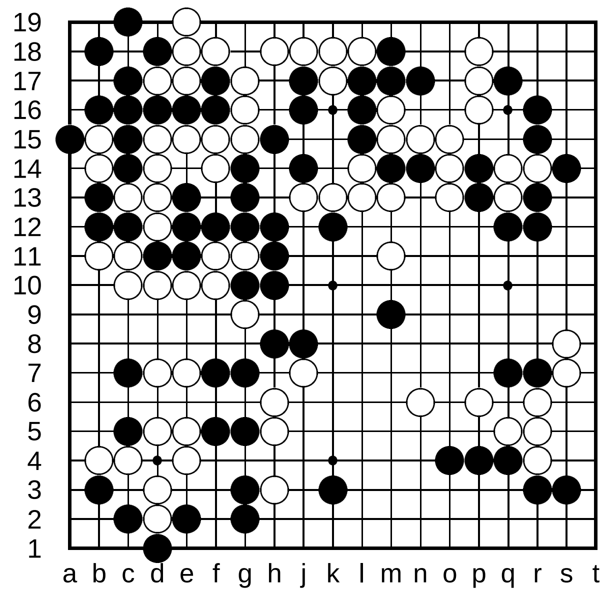

Diagram 2. A full board position represented by a dynamical system.

For example, the probability of the block with the stone on $r7$ to be captured is $1.0 - s_{r7}$ and is equal to the probability of the blocks with stones at $r4, s7$ not to be captured ($= s_{r4}s_{s7}$) and the points $p7, q6, q8, r8$ being occupied by White ($= w_{p7}w_{q6}w_{q8}w_{r8}$) giving relation (6).

Before the iteration all b_i and w_i variables are initialized to 0.5 and all s_j variables are initialized to 1.0 . Then the system is iterated until all values change less than some threshold parameter (for example 10^{-5}, its precise value is not crucial) at most some maximal number $k_{\max}$ of times (measurements reported in section VI-B use $k_{\max} = 5$).

V. Limits of what Dynamical Systems can do

Not all rules of Go are local by nature. The rule that players alternate in their moves sets limits to the usability of SEDS which are to be discussed in this section. On the other hand, in a local fight both sides may not alternate their moves. If the fight does not have highest priority then one side may not answer an opponent's move and play elsewhere. A different example of non-alternating moves is given in the appendix where a sacrifice move allows Black to move afterwords twice in a row in a crucial area.

A. Ladders

The action at a distance inflicted by ladder breaking stones seems to be a good example against local models like our dynamical system model. But as with plane waves in physics, with long distance effect being described through *local* differential equations one can not easily dismiss the possibility that ladder breaking stones could be described through a local model. However, ladders are a good counterexample against static evaluation. For example, to work out a long winding ladder like in diagram 3 statically - even if possible at all in principle - would be so much more difficult than simply performing the moves in a deep but narrow tree search.

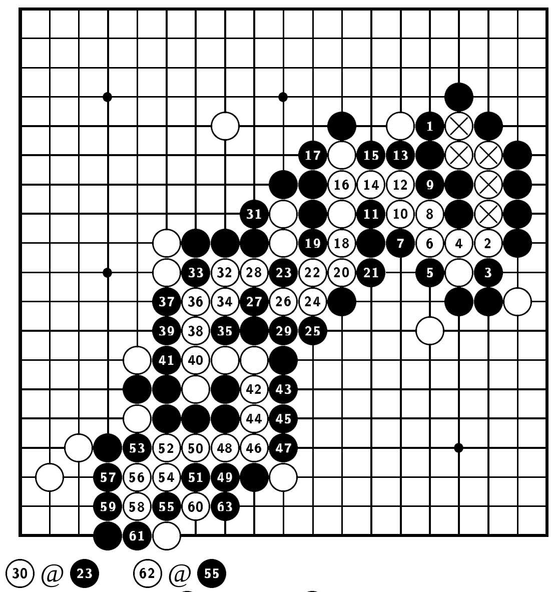

Diagram 3. ● catches ⊗ in a winding ladder.

B. Life & Death

An example for a concept in Go that is *not* a purely local phenomenon, i.e. it can not be described by considering only one block/point and its neighbours at a time, is the concept of *life* which is defined recursively: *A block is alive if it participates in at least two living eyes and an eye is alive if it is surrounded only by living blocks.* To identify unconditional life one has to consider the complete group of living blocks at once.

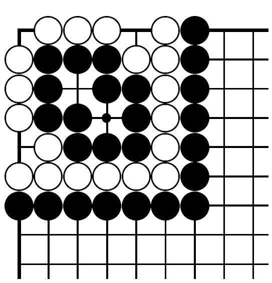
Diagram 4.

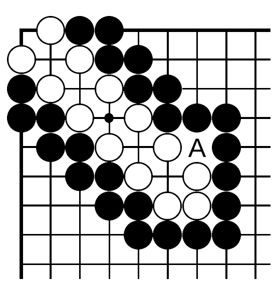

Diagram 5.

For example, the life of the white blocks in diagram 4 depend on each other and the conclusion that all blocks are alive can only be drawn at once, not in an iterative way and not by considering only one block and its neighbours at a time, so not by a purely local algorithm. Similarly, the life of the white stones in diagram 5 hangs on who moves next at the *distant* point A in a *discrete, non-iterative* way.

The current version of a SEDS computer program recognizes (non-local) static life (life without ever having to answer any threat) at the time when neighbourhood relations are established during the initialization of SEDS. Although this is a first step towards including life and death in SEDS, static life happens only rarely in games.

VI. Results

A. Existence, Uniqueness and Stability

The dynamical systems formulated along the lines of the previous section always have at least two solutions: one solution where all white blocks live, all black are dead and all points are fully under white influence and the same with switched colours. If all w_i, b_i, s_j are initialized according to one of these solutions, the iteration will keep these values stable.

In addition to these solutions in any board position computed so far the dynamical system had exactly one other solution (i.e. a fixed point of the dynamical system) with all values in the interval 0..1. This solution was obtained from any initial conditions other than the ones leading to the two extreme solutions mentioned above.

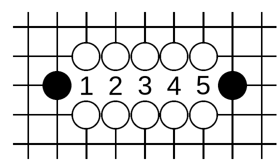

Diagram 6.

For example, when extending the position of diagram 1 to the one in diagram 6, using just for simplification the symmetry to identify points 2 and 4 and points 1 and 5, assuming that all stones are alive (i.e. $\bar{b}_1 = \bar{w}_1 = \bar{w}_2 = \bar{w}_3 = 1$) and using formula (4) repeatedly then the system of equations can be boiled down to one equation of degree 3 for w_2: $0 = 4{w_2}^3 - 2{w_2}^2 - 7w_2 + 4$ which has numerical solutions 1.259, 0.589, -1.348, only one of which is in the interval 0..1. But that is no lucky coincidence. Equation (4) guarantees that the single value for w_i lies in the interval $0 \le w_i \le 1$ if $\bar{b}_i, \bar{w}_i$ are in the interval $0 \ldots 1$. Furthermore, the case $\bar{b}_i = \bar{w}_i = 0$ is not possible.

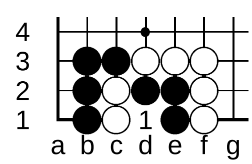

Diagram 7.

The following position in diagram 7 has three fixed points of which the meaningful one is stable. We assume that the surrounding blocks are alive, i.e. $s_{c3} = s_{d3} = 1$. The probability of the point $d1$ to have only black neighbours is $s_{d2}(1 - s_{c1})$ and thus the probability $\bar{w}_{d1}$ to have at least one white neighbour is $\bar{w}_{d1} = 1 - s_{d2}(1 - s_{c1})$. Similarly, we get $\bar{b}_{d1} = 1 - (1 - s_{d2})s_{c1}$ which inserted into equation (4) gives

$$w_{d1} = \frac{s_{c2}s_{d2} - s_{d2} + 1}{2s_{c2}s_{d2} - s_{c2} - s_{d2} + 2}.$$

The probability for the block at $d2$ to be captured is $s_{c1}w_{d1}$. Thus

$$s_{d2} = 1 - s_{c1}w_{d1}$$

and similarly

$$s_{c1} = 1 - s_{d2}b_{d1}.$$

Together with $w_{b1} + b_{d1} = 1$ this is the system to be solved. It has the 3 solutions

$$b_{d1} = w_{d1} = \frac{1}{2}, \quad s_{c1} = s_{d2} = \frac{2}{3} \ ;$$

$$b_{d1} = 0, \quad w_{d1} = 1, \quad s_{c1} = 1 \quad s_{d2} = 0 \ ;$$

$$b_{d1} = 1, \quad w_{d1} = 0, \quad s_{c1} = 0 \quad s_{d2} = 1$$

of which the first is stable and the others are the extreme solutions mentioned above. What seems to be inaccurate in the first solution is $s_{c1} = s_{d2} = \frac{2}{3}$ instead of $\frac{1}{2}$.

In the computation of $\bar{w}_i, \bar{b}_i, s_j$ we made the following systematic error. If two events A, B have probabilities p_A, p_B then the probability that A *and* B occur is equal p_Ap_B only if A and B are independent. In the above computation we said that the probability for the block at $d2$ in diagram 7 to be captured is $s_{c1}w_{d1}$. But if White plays on $d1$ then the strength of the white block on $c1$ increases and is not anymore s_{c1}, so both probabilities s_{c1}, w_{d1} are not independent.

A more clear cut example is shown in diagram 8 where the external blocks with stones on $d5, e4$ are assumed to be alive. The probabilities w_{c1}, w_{c2} for White to move on any one of the two points $c1, c2$ is initialized to 0.5 as for all moves and it stays > 0 during the iterations. But the real probability for White to go on both points in a game is zero due to the suicide rule in Go.

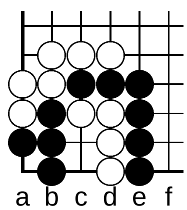

Diagram 8.

Although the systematic error of ignoring correlations between moves seems grave, the resulting numerical error is not. SEDS obtains $s_{b1} = 0.98, s_{d1} = 0.53$ which (given the scale of values obtained by SEDS in races of liberties) is a clear indication that the black block is alive and the white block is dead. This is a good example showing that SEDS is a robust method providing suitable indicators and data for more specialized semeai and life & death investigations.

A good question is how many iterations are necessary for all values to settle down so that all changes are less than some threshold parameter, say 10^{-5}. The interesting result is that for clear cut situations only few iterations (< 10) are necessary whereas for very unstable situations (for example, two attached blocks both under atari) the number of iterations can reach hundreds or thousands. This opens the possibility to get as a by-product a measure of instability. On the other hand determining unstable regions this way requires more iterations than would be needed for only a strength estimate.

In defence of the approach it should be mentioned that when establishing a ranking of moves often not the absolute strength values matter but the relative values.

B. *Statistics on Professional Games*

The SEDS function has been tested in detail by trying to predict the next move in professional games. Alternatively one can also look at these tests as tries to exclude as many moves as possible apart from the professional move if the main intention is to use SEDS to narrow the search of MCTS.

The test consisted of doing a 1-ply search for all positions occurring in all 50,000 professional games from the GoGoD collection [13]. For each board position in these games this includes

- performing each legal move in this position,
- iterating the dynamical system in the new position until the system stabilizes,
- adding up all probabilities of blocks to survive and points to be owned by either side and thus reaching a total score,
- ordering all legal moves according to their total scores,
- finding and recording the position of the professional move in the ranking of all moves.

The statistics have been recorded separately for each move number because at different stages of the game the static evaluation has different strengths and weaknesses.

The results of this test are reported under `http://lie.math.brocku.ca/twolf/papers/WoSE2010/2` due to the size of the diagram files. The data have been produced by evaluating 5.4 million positions from the 50,000 professional games of the GoGoD collection ([13]). On this web site three sequences of diagrams are shown. Each sequence contains over 400 diagrams, one for each move number. Figure 1 is one of the diagrams of the first sequence. It shows the number of positions in which the professional moves made in those positions land in the range $x \ldots (x+1)\%$ of legal moves as sorted by the SE where $x = 99$ are the top moves falling in the $99 \ldots 100\%$ range and $x = 0$ are the worst moves. Thus the higher the graph is on the right and the lower it is on the left, the better is the static evaluation.

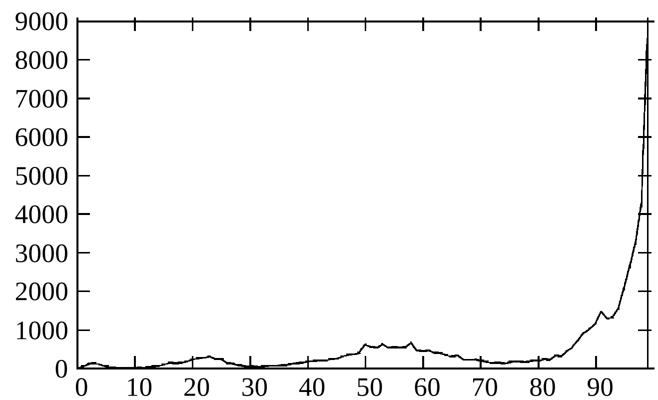

Fig. 1. A statistics of the ranking of the next professional move according to SEDS in all positions with move number 50 from 50000 professional games

The second series of diagrams differs from the first by having a logarithmic vertical axis. This is useful if the emphasis is to safely ignore moves from further consideration in MCTS because then we want to be reasonably sure that SE does not rank good moves (moves played by the professional player) as bad (on the left side of figure 1). In other words, we want to be sure that the graph is low on the left and to highlight that range a logarithmic vertical scale is useful. Figure 2 is the logarithmic version of figure 1).

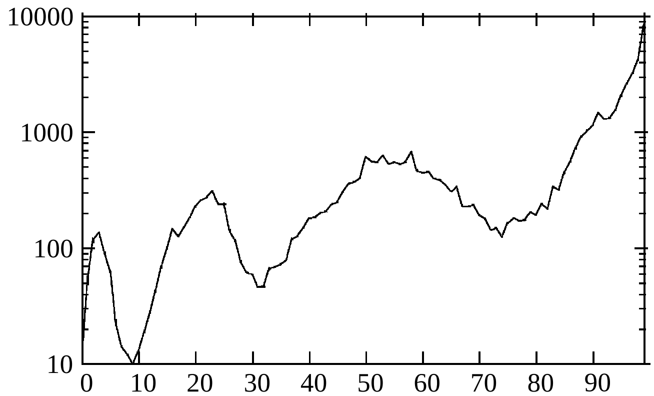

Fig. 2. The diagram of figure 1 here with logarithmic vertical axis

If one normalizes the vertical axis in figure 1 then this curve is the probability density $P(x)$ of the ranking of the professional move among all moves. If one accumulates this density from the right one obtains a so called 'survival function' $R(x)$: $R(x) := \sum_{u=x}^{99} P(u)$ which is displayed in figure 3. For example, a point on the graph with horizontal coordinate 85 and vertical coordinate 62 means: The professional move is kept with a probability of 62% (it survives) if the worst 85% of the moves are dropped (worst according to the SEDS).

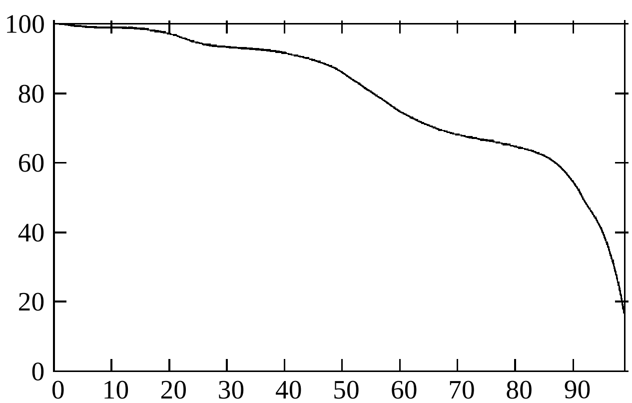

Fig. 3. The data of figure 1 here in a cumulative form of a survival function

C. Interpretation

In view of the simplicity of the static evaluation function SEDS the results as shown in figure 1 are surprisingly good.

Deficiencies are not difficult to explain. Because of the local concept of the dynamical system approach the SEDS has no concept of life (except a hard-wired fast recognition of static life), i.e. it does not know of the need of two eyes and the benefit of destroying eyes. For the SEDS in its current form (Dec 2009), strength is 100% correlated with resistance against being captured. As a consequence, sacrifice moves, like Black on A in diagram 9, get a low ranking. This is an extreme example where the professional move (Black on A) gets the lowest ranking of all moves by SEDS. Moves of this type make up the leftmost hump in figure 2.

Whereas the humps on the left of figure 2 are due to good (professional) moves getting a low evaluation by SEDS, the dents on the right of the graph are due to bad moves getting a high evaluation by SEDS. The rightmost dent in figure 2 is due to the feature of the evaluation function to favour moves on the 2^{nd} line, especially the 2-2 points (e.g. on a 19×19 board the points b2,b18,s2,s18). Again, this is a consequence of not knowing about the need for 2 eyes due to not knowing that Black and White can not do 2 moves at once, i.e. fill 2

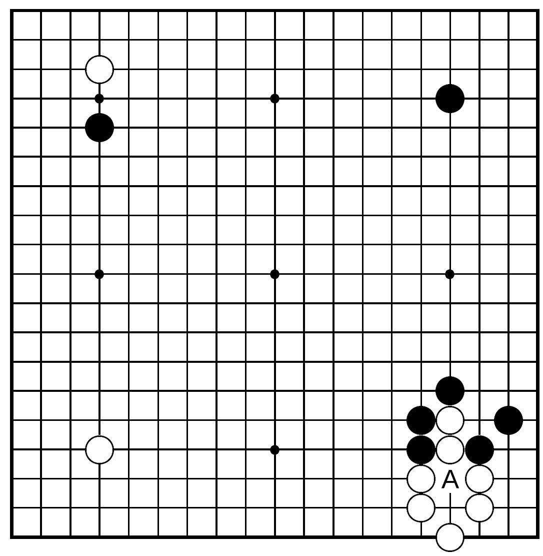

Diagram 9. A position from a professional game where SEDS fails due to lacking a concept of eyes.

eyes at once. For the SEDS a move on a 3^{rd} or 4^{th} line can be cut under by the opponent on the 2^{nd} line because SEDS does not know that the move on the 2^{nd} line needs two eyes.

When skipping through diagrams on `http://lie.math.brocku.ca/twolf/papers/WoSE2010/2` one sees that the current version of SEDS is most useful early in the game when life & death fights do not play a big role yet but also after about 15 moves when the exact influence of the edge of the board is not so crucial anymore.

Based on these findings it is expected that an appropriate consideration of life based on 2 eyes when computing the strength of blocks will lead to a significant improvement of SEDS. The problem is to find a natural merger of the need for 2 eyes with the current interaction formula (4). Also, the concept of life is not local, so the solution of the dynamical system and the determination of a (non-local) measure of life based on 2 eyes must be merged naturally into one algorithm. Of course, one could make quick progress with a superficial repair but the aim of this exercise is to get a lasting concept that has no artificial parameters and no artificial constructs and thus scales (can improve indefinitely with increasingly available computer power) and thus has the potential to result in a strong program in the long run.

D. Timing

The following times have been recorded on a Dell Optiplex GX620 PC with Intel(R) Pentium(R) D CPU 3.40GHz processor with 2MB cache size running Linux. One CPU was used. Times reported in table 1 are the result of averaging the times for evaluating 400 positions for each move number given in the table, each position from a different professional game from the GoGoD game collection [13].

The computations in column 2 include making a move and updating the strength of blocks and influence at points on the whole board. In column 3 these steps are made for each legal move.

move number	time in μs for static evaluation	time in ms for ranking moves
10	46.2	16.2
30	51.8	17.1
100	73.8	19.2
130	83.9	19.3
200	104.4	16.7
300	180.0	10.8

Table 1. Average times for static evaluation and ranking of moves.

As the update of the strength of a block is slower to compute than an update of the influence at a point, the static evaluation (column 2) becomes slower as more stones are placed on the board. On the other hand, the more stones there are on the board, the fewer legal moves exist and ranking all moves (column 3) becomes faster.

E. Parameter Dependence

The current form of SEDS has two parameters. One parameter `stop_value` is the minimal change of influence and strength which keeps the iteration alive, i.e. if the influence in a point or the strength of a block changes by at least this amount then the neighbouring points and blocks have to be iterated again. The other parameter `max_iter` is the maximum number of iterations for any point or block. When testing `stop_value` $= 10^{-2...-7}$and `max_iter` values $= 2 \dots 10, 100, 1000$ the following was found.

- The `max_iter` value has hardly any effect, neither on time, nor on accuracy because the number of iterations is typically very low (≤ 5). Rare exceptions occur in beginner games when several blocks with only one liberty are neighbour to each other and thus the situation is extremely unstable and many iterations between such blocks would occur.
- A `stop_value` of 10^{-3} gives the best performance/cost ratio in predicting moves in professional games. Lowering this value to 10^{-5} adds another 25% to the computing time and lowering it to 10^{-7} adds again 15% without improving the prediction of professional moves noticeably. Increasing the `stop_value` to 10^{-2} gives only very little time savings but lowers performance.
- An important conclusion is that allowing at least 3 or 4 iterations per item (point or block) does improve predictions. This is an indirect proof that SEDS recognizes, for example, when a white block is strong because a neighbouring black block is weak because a neighbouring white block is strong.

To summarize, `max_iter` is a parameter in the program but not a parameter for the computation that strikes a balance between accuracy and complexity. `stop_value` has an effect on accuracy and complexity but only a minor one. It is a 'natural' parameter in contrast to what one may call 'artificial' parameters, like

- the size of fixed size pattern that are used,
- ad hoc parameters that classify the strength of blocks like a threshold number of liberties such that a block is considered alive,
- a fixed size distance of the last move which favours follow up moves in its neighbourhood.

The dependence of accuracy and complexity of computation on such parameters is very uneven. If one would vary such a parameter and plot the resulting accuracy versus the computational complexity of the calculation then the curve would be flat and rise very slowly for low and medium complexity and would only for very large computational complexity rise more steeply and reach higher accuracy.

In contrast, 'natural' parameters have a more direct relation to the accuracy of the computation, like

- parameters regulating a genetic learning algorithm, or the speed of lowering the temperature of simulated annealing,
- the number of terms of an Fourier or Taylor series expansion of a function,
- the number of simulations of MCTS,
- the `stop_value` parameter for iterations of SEDS.

VII. Comparisons with other static evaluations

A. Criteria for Static Evaluations

Because there is ambiguity of what one can consider as an evaluation function, there are many criteria that matter for their characterization. An evaluation function is good if it

1) ranks top and good moves high,
2) ranks bad moves low,
3) is fast,
4) is flexible in time management, i.e. it is able to find reasonable quality moves if only little time is available and better moves if more time is available,
5) knows about the value of the next move in order to take more time and be more accurate if much is on stake,
6) knows when its own predictions are less accurate (e.g. in an unstable region where search is the proper tool) or more accurate (in a stable region),
7) knows about the risk involved in a move to select safer moves when being ahead and more risky moves when being behind,

8) knows about the strengths of the search program and thus gives moves a high priority if they lead to positions which the search can handle well,
9) performs equally well for positions resulting from skillful play (professional games), resulting from beginner play or from high handicap games,
10) provides, as a side product, data about points and blocks on the board, date which are re-usable in the computation of higher concepts,
11) has potential for improvement, e.g. if extensions can be added efficiently, for example, the recognition of safe links or of more general semeai.

B. Comparisons

The first ability is easily tested by predicting moves of professional games. Figure 4 compares SEDS with other programs.

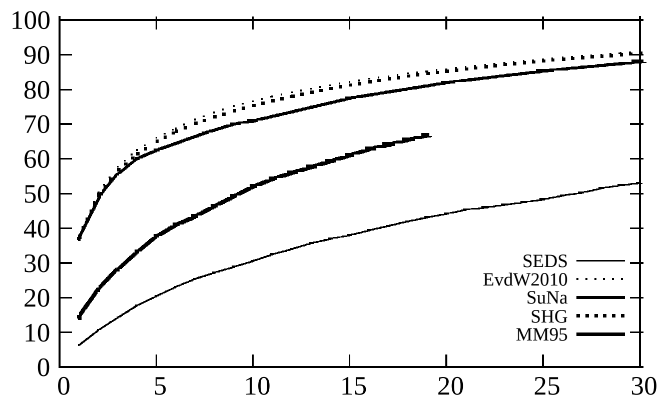

Fig. 4. A statistics of the ranking of the next professional move by different evaluation functions

The curves EvdW10 ([14]), SuNa ([15]), SHG ([16],[17]) are close. They are produced by programs that match pattern learned from professional games but newer versions match also other features (see [17]). Such programs are the best in predicting professional moves, especially in the opening. They are also reasonably fast with typically 0.3-100 ms for a full board ranking. But because of weaknesses in criteria 4-10 and partially 11 they are not suitable for playing complete games. Instead, pattern matching programs are useful to provide prior knowledge to MCTS especially in the opening, as applied, for example, in the program Steenvreter from Erik van der Werf ([14]).

For criterion 2 no tests are currently available. It would be interesting to see how pattern matching programs perform in positions resulting from beginners play which may show only small patterns learned from professional games and which consequences this has on the evaluation of bad moves.

It may be unexpected that SEDS, which on a first look seems to be extremely slow, is in fact comparable to or even faster in speed than SuNa ([15]) and of same speed as the pattern matching programs described in [18], [17], although slower than EvdW10 ([14]) (according to personal communication with the authors). In section 3.4 of [19] it is commented that the time for computing the heuristic value of a move in the program MANGO (mainly consumed by pattern matching) is in the order of 1 ms which would be about 10 times slower than the times shown in column 2 of Table 1.

In predicting professional moves SEDS is clearly weaker than the pattern matching programs but its strength is that it has good potential for criteria 6) (the local number of iterations turns out to be a good indicator for the stability of a block or region), 7) (because SEDS has a numerical value for a score and a measure of instability), 9) (SEDS is not based on pattern learned from a specific class of positions), 10) (SEDS generates and stores all neighbourhood relations because it needs them frequently and it provides influence values at points and strength values of blocks), and 11) (higher concepts are easier to be built on top of rich available basic data).

The graph MM95 in figure 4 is the cumulative version of the figure in section 8.2 of Martin Müller's PhD thesis [20] produced by the program Explorer. It is an elaborate knowledge based program [21], thus it naturally scores better in move prediction than SEDS which is based purely on neighbourhood relations. It would be interesting to compare the time consumption of the Explorer evaluation function and SEDS.

Another interesting study would be to simulate an evaluation function by a MCTS based program that is given only 1-10 ms time. MCTS has advantages in criteria 4, 7, 8, 9 and partially 10, 11 compared to pattern matching programs.

C. Characterization of SEDS

The dynamical system as formulated in section IV is a first version that allows us to study general properties of such an approach. This version is minimal and has no concept of 100% safe liberties (eyes) and thus has no concept of life in general[2] only a concept of strength as the resistance against being captured and a concept of influence at empty points. As move prediction experiments show, these concepts already capture a good part of the nature of Go.

The value of SEDS lies in providing a new source of Go knowledge that can be experimented with and potentially used to create prior knowledge for MCTS programs. The following are the strengths of SEDS in its current form.

- The provided knowledge is of global nature because all points and blocks are coupled to their neighbours and so on. For example, one block 'feels' the weakness of an opponent neighbouring block due to the strength of an own block behind that.
- SEDS is applicable to any board position and it is robust in the sense that approximations in the assumptions, like the non-correlation of future moves, do only disturb numerical results, and do not make them completely wrong.
- In all tested positions iterations lead to a single fixed point of the dynamical system which describes the strengths of

[2]As a start, static life which never has to answer any move has been implemented.

all blocks and the influence at all points. The fixed point reflects the situation on the whole board, no artificial local cutoff is assumed which is, for example, the case when local pattern are used.
- The computation of fixed points is reasonably fast because only few iterations are needed and an incremental form of the computation is possible which limits iterations to areas where a move that has been made has an effect on its neighbourhood. Consequently, the speed of SEDS is comparable to that of pattern matching programs and thus probably faster than traditional knowledge based programs. The ranking of moves based on the SEDS scores they reach, can be parallelized and a more heavy SEDS (e.g. by including life & death computations) makes the parallelization more coarse grain and more effective.
- The influence radiated from blocks depends on their strength and the block's strength depends on the influence at neighbouring points. None is artificially fixed.
- The special situation of points and blocks at the boundary is taken care of automatically due the naturally reduced number of neighbouring points and blocks.
- The SEDS described in this paper is pure in the sense that it is free of artificial parameters and thus has potential to be extended and tuned.
- Individual components of the SEDS can be modified independently, like the dependence of the strength of a block on the influence of neighbouring points or on the strength of neighbouring blocks or like the dependence of influence at a point on the strength of neighbouring blocks or influence at neighbouring points.
- As a by-product of SEDS information on the strength of blocks and the influence at points is obtained that can be useful for other specialized modules, like tactical search or generalized strength pattern.
- Weaknesses of SEDS can be determined easily by doing move prediction experiments and filtering out positions where the professional move ranks low in the SEDS ranking.

VIII. Future Tasks

Three directions of future work are: to improve the computation of the strength of blocks and the propagation of influence, to make good use of the computed values and to merge static evaluation with MCTS.

A. Improvements of the Algorithms

The following are possibilities to improve this first version of SEDS.

- The formula for the strength of a block could be improved by giving the number of liberties a weight in addition to the currently used influence values at the liberties.
- An inspection of worst performances in move predictions showed that the concept of life, i.e. the need for two eyes is not automatically covered by this first version of SEDS and needs to be added. A possibility would be to perform iterations in two stages. After a first sequence of iterations settled down a quick life and death analysis using available influence values could be performed and resulting strength values of blocks be considered in a second series of iterations.
- Another non-static concept based on moves are safe links. These are also not automatically covered by SEDS and have to be incorporated in some organic way (see section VIII-B below for a suggestion).

  The reason for not including eyes and links already into this initial version of SEDS was to study at first a pure version of dynamical system before extending it. The purpose of minimizing the number of concepts and parameters is to keep the program scalable in the long run.
- SEDS should provide a local awareness of stability (i.e. the dependence of the local strength measures on who moves next), of the size of an unstable area and thus of the importance of investigating moves in this area. If this measure proves to be a good heuristic for the urgency to investigate moves through MCTS then the 1-ply search to rank moves would not be needed anymore. This would result in a significant speedup.
- Because pattern matching and SEDS are complimentary in their approach, a combination of both has good chances to be better than each one of them.

  In pattern matching the shape of blocks matters whereas the strength of blocks touching the pattern edge and potentially extending beyond the edge is unknown. Differently in SEDS, shapes of blocks are ignored but approximate strength values are known. Pattern are local objects whereas strengths and influences computed by SEDS are global objects.

  In a first approach of merging pattern matching and SEDS one could simply add both move recommendations with appropriate weights. The next stage would consider pattern enriched with strength and influence values. Such pattern could be smaller and still have high predictive value.

B. Multipole Moments of Influence

In this section ideas are given how to utilize an influence field.

The philosophy behind SEDS is to model the 'real strength' of blocks and 'real influence' as close as possible to what strong human players think about that. If the resulting influence field would be a proper model to describe what is going on on a Go board (i.e. if it would have some intrinsic meaning for Go) then to some extend also the components of a multipole expansion of that field (i.e. the discrete analogue of that) should have a meaning in Go. Multipole moments describe changes of a field in the neighbourhood of a point.

The first term of such an expansion (monopole component) is the value of the field itself in a point. It is an indicator of the likelihood of a point to be owned by either side at the end of the game.

The second term (dipole moment, for continuous fields known as gradient) indicates the direction of fastest increase of influence and the rate of change of influence in this direction and could be used as indicator how to reach or avoid a strong block of a specific colour, or how to run out in the open to gain liberties.

The third term (quadrupole moment) contains second order differences. What is of interest from these is a measure of how much the point is a saddle point and can be used as an indicator how important the point is for connecting or cutting blocks. For example, in Diagram 10 let A, B, C, D be influence values at these points (or strength values of blocks with a stone at these points), all being neighbours to the point X then $|A + D - B - C|$ is a good indicator for the urgency to connect own blocks and/or separate and cut opponent blocks. The formula could be refined by including diagonal neighbours.

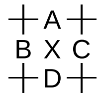

Diagram 10.

C. Merging Static Evaluation and Treesearch

Whereas MCTS is self-sufficient for game playing, evaluation functions are not. When using SEDS to make moves directly to play games as program `moag-0.3` on CGOS ([22]) all games were lost with the exception of two wins against `AmiGoGtp`.

The merger of MCTS and an improved SEDS will be a main future task and a research project on its own. In doing this one would want to be able to vary smoothly the times allocated to both, at first statically then dynamically depending on the board position.

The computational cost of SEDS is more justified in the opening when precise MCTS is very expensive and in non-fighting positions when SEDS is more accurate. Investing time in SEDS will be less justified towards the end of the game when MCTS does a perfect job and in all-or-nothing fights when SEDS is not accurate enough due the occurrence of sacrifice moves where the shape of throw-in blocks matters which is disregarded in SEDS. As a first attempt one could add recommendations for moves from MCTS and SEDS as described in formula (3.2) in [19].

A run-time library under Linux is available that provides influence values at points and strength values of blocks. It also has a function which ranks all legal moves by their estimated quality. To try it out within a MCTS program please contact the author.

IX. Summary

Based on the local nature of the capture rule in Go, which is currently not taken advantage of in MCTS, it is argued that a dynamical system (DS) is suitable to describe a board position in Go. It is shown how minimal natural assumptions lead to the formulation of a DS that has a number of useful properties.

After a discussion of principal limitations of a local dynamical system approach the results of move prediction tests in professional games are described.Given the simplicity of the currently investigated SEDS not using any Go-knowledge from high level games, no pre-computed information, no tactical information or algorithms other than static life and death its move prediction abilities are surprisingly good. In comparisons with other static evaluations strengths and weaknesses of SEDS are described.

The characteristics of SEDS and the many possible improvements listed in the paper give the new approach much potential that should be explored.

Appendix

In this appendix an example is given to support the claim that in Go there is a field that embodies strength and influence, which is not an artificial human construct but which is at the heart of the game and is of intrinsic, fundamental nature. In the following position such a hypothetical field is used to explain the optimal move which is a sacrifice.

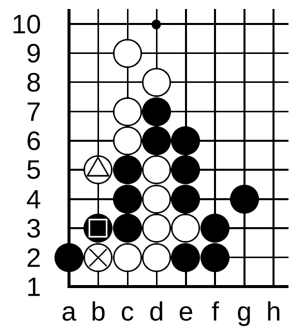

Diagram 11.
● to move.

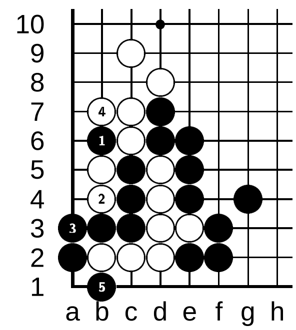

Diagram 12.
● lives.

In diagram 11 the aim of Black is to make its corner block ◙ alive which can only happen by capturing one of the white blocks △ , ⊗ . But neither one can be captured directly by playing only in its vicinity (● on b6 would be captured by ○ on b7 and ● on the first row would be answered by ○ on b4 and would be too slow). Nevertheless, a static evaluation function modeling an influence field, for example the one described in section IV, would give small but nonzero values for Black's strength around a6, b6 and on the first row at b1, c1, d1. If modelled correctly these influence values alone should be too low to indicate the death of one of the white blocks ⊗, △ individually but if all influence is added up and increased by a value equivalent to the advantage of having the turn then the total should be enough to indicate life for ◙ .

The remaining question is how to convert this prospect of life for Black into a good first move. The influence field also helps here.

If this field is to be meaningful then it should not change erratically from one move to the next, except at the very end when the position becomes completely settled and the value jumps to one of the two extremes. So Black can not expect to change the total sum of all Black strengths through a clever move. But what Black can expect to do is to shift the distribution of its influence. To succeed in this example, Black needs to bundle all its influence onto its weakest white neighbour block, which is ⊗ , i.e. to move its small influence at a6, b6 towards the block ⊗ to weaken it further. The result of the sequence in diagram 12 is that White now has total control of the points a6, b6 and Black in exchange gets one extra move towards capturing the block $\otimes$ which is enough in this case.

This principle of bundling influence to overcome a threshold strength in a local target area in order to live, link, kill or cut, explains all sacrifice moves, not only in this example and not only in life and death problems. This is a good guidance not only for players but also for computer programs to formulate tactical and strategic aims, on one hand being bound by a stable total sum of influence but on the other hand being allowed to shift influence and to focus it to overcome threshold values locally for living and killing. These threshold values result from the discrete requirement of having two eyes for life and the discrete nature of the capture rule of capturing all stones of a block at once.

A variation of this principle is to look for moves which make own weak stones good sacrifice stones, aiming to move the remaining strength of these stones to a distant and more important area.

In some games the principle of collecting thinly spread potential through playing 'light', i.e. through playing stones which have a good chance of being sacrificed later, is not only a tactical concept but a strategic one. When a professional player gives a strong amateur player many handicap stones then Black can not simply be fooled, the only way for White to win is to collect all potential on the board by playing light and probably sacrificing stones.

But also in even games thinly spread influence/potential is an important concept. Influence may be shifted around to force the opponent to become very strong on one side of a local area in order to gather own strength and be better prepared to attack on the opposite side of that area, in other words to create imbalance in the opponents position. This is known as a proverb: Attach against the stronger stone (for example, in[23], p. 121).

ACKNOWLEDGEMENTS

Work on this project was supported by a DARPA seedlings grant. The author thanks the other members of the funded group from the Stanford Research Institute for discussions, especially Sam Owre for binding SEDS into the Fuego program and running tests on CGOS. Bill Spight is thanked for comments and discussions. Computer tests were run on a Sharcnet cluster `http://www.sharcnet.ca`.

REFERENCES

[1] B. Brügmann, "Monte Carlo Go," 1993, preprint `ftp://ftp.cgl.ucsf.edu/pub/pett/go/ladder/mcgo.ps`.

[2] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Machine Learning*, vol. 47, no. 2/3, pp. 235–256, 2002.

[3] L. Kocsis and C. Szepesvári, "Bandit Based Monte-Carlo Planning," in *Machine Learning: ECML 2006, Lecture Notes in Artificial Intelligence 4212*, M. S. J. Fuernkranz, T. Scheffer, Ed. Springer, 2006, p. 282293.

[4] B. Bouzy and B. Helmstetter, "Monte-Carlo Go developments," in *Advances in Computer Games. Many Games, Many Challenges. Proceedings of the ICGA / IFIP SG16 10th Advances in Computer Games Conference*, J. van den Herik, H. Iida, and E. Heinz, Eds. Kluwer Academic Publishers, 2004, pp. 159 – 174.

[5] S. Gelly, "MoGo home page," `http://www.lri.fr/~gelly/MoGo.htm`.

[6] O. Teytaud, "MoGo: a software for the Game of Go," `http://www.lri.fr/~teytaud/mogo.html`.

[7] R. Coulom, "Crazystone home page," `http://remi.coulom.free.fr/CrazyStone/`.

[8] C. Lee, M. Wang, G. Chaslot, J. Hoock, A. Rimmel, O. Teytaud, S. Tsai, S. Hsu, and T. Hong, "The computational intelligence of MoGo revealed in Taiwan's computer go tournaments," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 1, no. 1, pp. 73–89, 2009.

[9] K. Chen, "Computer Go: Knowledge, Search, and Move Decision," *ICGA Journal*, vol. 24, no. 4, pp. 203–215, 2001.

[10] B. Bouzy and T. Cazenave, "Computer Go: An AI-Oriented Survey," *Artificial Intelligence*, vol. 132, no. 1, pp. 39–103, 2001.

[11] M. Müller, "Computer Go," *Artificial Intelligence*, vol. 134, no. 1-2, pp. 145–179, 2002.

[12] T. Wolf, "Positions in the game of go as complex systems," in *Proceedings of 2009 IEEE Toronto International Conference (TIC-STH 2009) - Science and Technology for Humanity*. Toronto, Canada: IEEE, 2009, pp. 222–229.

[13] T. Hall and J. Fairbairn, "GoGoD Database," 2007, CD-ROM with $50,000$ professional games, `http://www.gogod.co.uk`.

[14] E. van der Werf, "about steenvreter," 2010, personal communication.

[15] I. Sutskever and V. Nair, "Mimicking Go Experts with Convolutional Neural Networks," *ICANN*, vol. 2, pp. 101–110, 2008.

[16] D. Stern, R. Herbrich, and T. Graepel, "Bayesian pattern ranking for move prediction in the game of Go," in *ICML '06: Proceedings of the 23rd International Conference on Machine Learning*. ACM Press, 2006, pp. 873–880.

[17] D. Stern, "Modelling Uncertainty in the Game of Go," 2008, phD thesis, University of Cambridge, `http://research.microsoft.com/pubs/74404/thesis.pdf`.

[18] E. van der Werf, J. Uiterwijk, E. Postma, and van den Herik H.J., "Local move prediction in Go," in *Computers and Games: Third International Conference, CG 2002, Edmonton, Canada, July 2002*, J. J. Schaeffer, M. Müller, and Y. Björnsson, Eds. Springer Verlag, Berlin, 2003, pp. 393–412.

[19] G. Chaslot, M. Winands, H. van den Herik, J. Uiterwijk, and B. Bouzy, "Progressive strategies for Monte-Carlo Tree Search," *New Mathematics and Natural Computation*, vol. 4, no. 3, pp. 343–357, 2008.

[20] M. Müller, "Computer go as a sum of local games: An application of combinatorial game theory," 1995, phD thesis, ETH Zürich, `ftp://ftp.inf.ethz.ch/pub/publications/dissertations/th11006.ps.gz`.

[21] ——, "Counting the score: Position evaluation in computer Go," *ICGA Journal*, vol. 25, no. 4, pp. 219–228, 2002.

[22] D. Dailey, "Computer Go Server home page," `http://cgos.boardspace.net/`.

[23] O. Hideo, *Opening Theory Made Easy*. The Ishi Press, 1992.