Combinatorics of Go

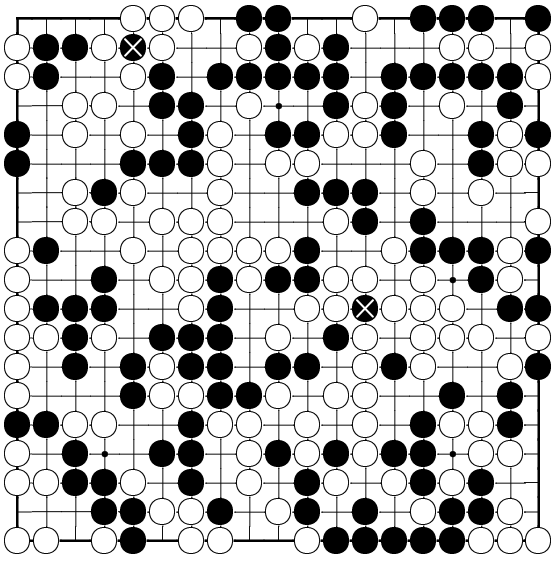
John Tromp Gunnar Farnebäck

January 31, 2016

Abstract

We present several results concerning the number of positions and games of Go. We derive recurrences for $L(m, n)$, the number of legal positions on an $m \times n$ board, and develop a dynamic programming algorithm which computes $L(m, n)$ in time $O(m^3 n^2 \lambda^m)$ and space $O(m\lambda^m)$, for some constant $\lambda < 5.4$. We used this to compute $L(n, n)$ up to the standard board size $n = 19$. In ternary (mapping 0,1,2 to empty,black,white)

$L(19, 19) =$



For even larger boards, we prove existence of a *base of liberties*

$$L = \lim_{m,n\to\infty} \sqrt[mn]{L(m, n)} = 2.975734192043357249381\ldots$$

Based on a conjecture about vanishing error-terms, we derive an asymptotic formula for $L(m, n)$, which is shown to be highly accurate.

We also study the Game Tree complexity of Go, proving an upper bound on the number of possible games of $(mn)^{L(m,n)}$ and a lower bound of $2^{2^{n^2/2-O(n)}}$ on $n \times n$ and $2^{2^{n-1}}$ on $1 \times n$ boards, in addition to exact counts for $mn \leq 4$ and estimates up to $mn = 9$. We end with investigating whether one game can encompass all legal positions.

1 Introduction

Originating over 3000 years ago in China, Go [2] is perhaps the oldest boardgame in the world, enjoyed by millions of players worldwide. Its deceptively simple rules [3] give rise to amazing strategic depth. Results about the computational complexity of Go date back some 35 years. In 1980, Lichtenstein and Sipser [8] proved Go PSPACE-hard, while 3 years later, Robson [10] showed Go with the basic ko rule to be EXPTIME-complete. More recently, certain subproblems of the game have been shown PSPACE-complete, like endgames [7] and ladders [14]. This paper focuses instead on the *state complexity* of Go. We are motivated by the fact that the number of legal positions is a fundamental property of a game, the notion of legal position being unambigiously defined for Go despite many variations in rulesets, and that its computation turns out to be a huge computational challenge which was only met in 2016.

2 Previous work

Results about the state complexity of Go have been mostly confined to the online newsgroup `rec.games.go` and the `computer-go` mailing list. In September 1992, a `rec.games.go` thread "complexity of go" raised the question of how many positions are legal. It was noted that a trivial upper bound is 3^{mn}, since every point on the board may be empty, black, or white. A position is legal if and only if every string of connected stones of the same color has an empty point adjacent to it. Achim Flammenkamp was the first to post simulation results, showing that $L(19,19) \sim 0.012 \times 3^{361} \sim 2.089 \times 10^{170}$. In August 1994, a thread "Complexity of Chess and Go" revisited the problem. Jack Hahn, Jonathan Cano, and John Tromp all posted programs to compute the number of legal positions by brute force enumeration. The largest count published at the time was $L(4,5) = 1840058693$. A week's worth of computation would have found $L(5,5)$ as well, but enumerating $L(6,6)$ takes over 10000 times longer, severely limiting this approach.

In a January 2000 thread "Number of Legal Positions on Almost Rectangular Boards", inspired by earlier remarks by John Tromp and Hans Zschintzsch, Les Fables first explained in detail how to count using dynamic programming. His remark "*Calculation for 9x9 should be possible on any PC, and a supercomputer should easily be able to handle 13x13.*" proves to be spot on. Much later, on January 23, 2005, Eric Boesch independently discovered this method on the `computer-go` mailing list. His method is implemented the next day by Tapani Raiko, but a bug leads him to post a wrong count for $L(5,5)$. Later that day Jeffrey Rainy, based on his own implementation, gives the correct values for $L(5,5)$ and $L(6,6)$ but wrong values for $L(7,7)$ and $L(8,8)$. Finally, the next day, Gunnar Farnebäck posts the first bugfree program, providing counts up to $L(10,10)$. In June 1999, a thread "Math and Go" discussed the number of games of Go. Robert Jasiek claimed an upper bound of n^{3^n}, which still needs to be corrected for intermediate passes. John Tromp showed how to get a double

exponential lower bound, which we make more formal while fixing a slight flaw, in this paper. In the same month, John Tromp started a thread "number of 2*2 games", noting that the number is 386356909593, as was recently independently verified.

3 History of compute intensive results

Results from Tromp's memory based C program were posted on June 29, 2005 for 13x13, and August 11, 2005 for 14x14, when Michal Koucký helped develop a file-based version to get around memory limitations, and introduced the use of Chinese Remaindering. That led to posting legal counts for 15x15 on August 28, 2005, for 16x16 on October 6, 2005 and much later for 17x17, on August 18, 2006. It wasn't until early 2014 that Tromp got an offer from Piet Hut at the Institute for Advanced Studies, to use their computing cluster for 18x18. These were announced on Hacker News on March 9, 2014 accompanied by a request for yet more computing power to tackle 19x19, that was finally announced on Jan 22, 2016.

4 Preliminaries

A position on an $m \times n$ Go board is a mapping from the set of *points* $\{0, \ldots, m-1\} \times \{0, \ldots, n-1\}$ to the set of colors $\{\text{empty}, \text{black}, \text{white}\}$. Points are *adjacent* in the usual grid sense—equal in one coordinate and differing by one in the other. A point colored black or white is called a *stone*. Adjacent stones of the same color form connected components called *strings*. An empty point adjacent to a string is called a *liberty* of that string. A game of Go starts with an empty board. The players, black and white, alternate turns, starting with black. On his turn, a player can either *pass*, or make a move which doesn't repeat an earlier position. This is the so-called Positional SuperKo (PSK) rule. Some rulesets, notably the American Go Association's AGA rules, use the Situational SuperKo (SSK) rule, which only forbids repeating a position with the same player to move. A move consists of coloring an empty point your color, followed by emptying all opponent strings without liberties (capture), followed by emptying all your own strings which then have no liberties (suicide). Finally, two consecutive passes end the game [1]. Thus, in positions arising in a Go game, strings always have liberties. Such positions are called *legal*. The number of legal $m \times n$ positions is denoted $L(m, n)$.

Figures 3 and 4 show all positions on a 2×2 board. Obviously, the 16 positions with 4 stones are illegal. Additionally, the 8 positions with a stone of one color neighboured by two stones of the opposite color, are illegal. Since all other 3 stone positions and all positions with 2 or fewer stones are legal, we find that $L(2, 2) = 3^4 - 16 - 8 = 57$.

[1] since none of our results is concerned with the outcome of a game, we refrain from discussing scoring rules, except to note that these account for the major variation in rulesets

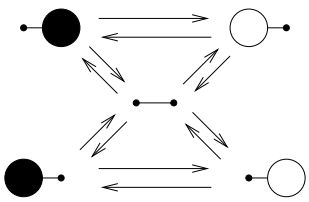


Figure 1: game graph $G(1,2)$.

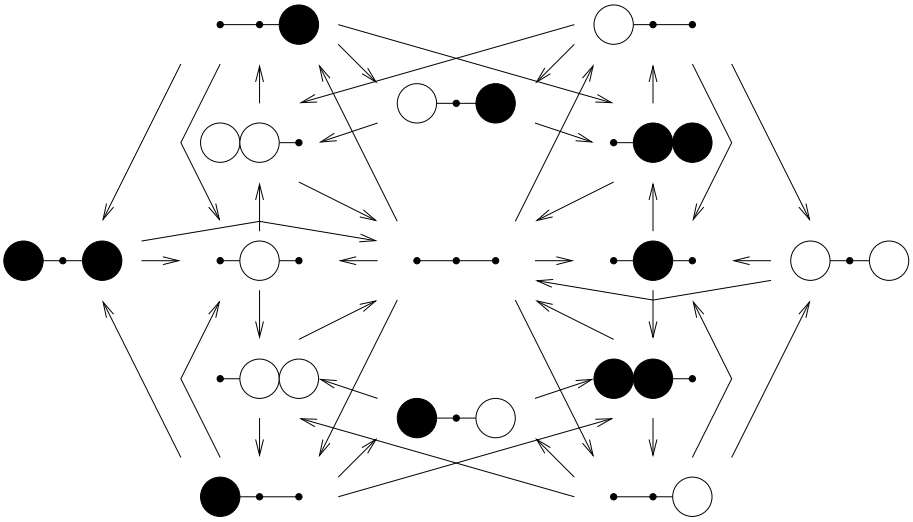


Figure 2: game graph $G(1,3)$.

DEFINITION 1 (GAME GRAPH) *Let $G(m,n)$ be the directed graph whose vertices are the legal $m \times n$ positions, and which has a directed edge $p \to q$ whenever a white or black move from position p results in position $q \neq p$.*

Note that we exclude self-loops, corresponding to single-stone suicides, which are prohibited by the PSK rule. This will prove useful in Lemma 1 below.

Figure 1 shows $G(1,2)$, consisting of 5 nodes and 12 edges, Figure 2 shows $G(1,3)$, consisting of 15 nodes and 42 edges, while Figure 4 shows $G(2,2)$ factored into rotation/mirror symmetry classes. Piet Hut observed that these latter two graphs are the only ones with no 2-loops. We next establish some basic properties of Go game graphs.

LEMMA 1 *Outgoing edges from a position are in 1-1 correspondence with moves that are not single-stone suicides.*

PROOF. Given a position p, each move uniquely determines a resulting position q and hence an edge $p \to q$. It remains to show the converse; that

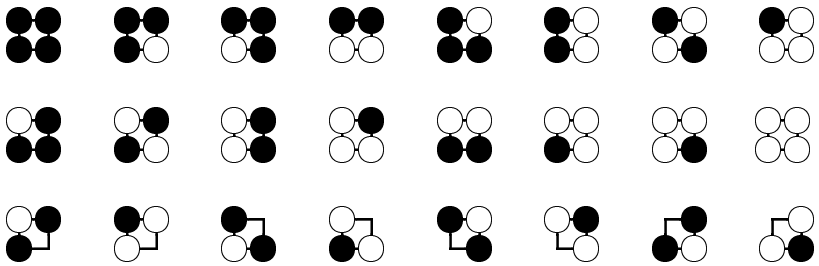


Figure 3: All $16 + 8 = 24$ illegal 2×2 positions

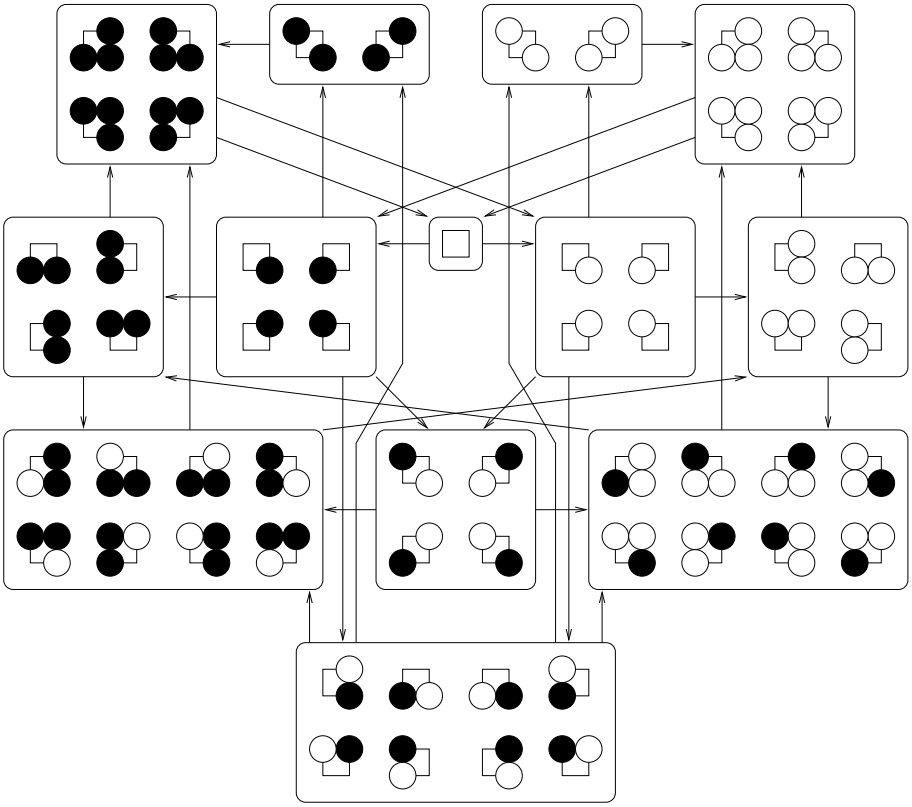


Figure 4: game graph $G(2, 2)$ containing all 81-24=57 legal positions, with nodes/edges grouped into rotation/mirror symmetry classes.

a resulting position $q \neq p$ uniquely determines a move. If q has one more stone of color c, say at position (x, y), and the same or fewer stones of opposite color, then that was the move. If q has fewer stones of color c and the same number of stones of opposite color, then the missing stones must form a string with 1 liberty, and the move was a suicide. It can be seen that these cases are exhaustive, the former covering all non-suicide moves, and the latter covering all multiple-stone suicides. •

这里的推论和引理
基本都是比较显然的

COROLLARY 1 *A node with k empty points has outdegree at most $2k$.*

COROLLARY 2 *Each edge has an implied black or white color.*

Recall that a *simple* path is one that has no repeated vertices.

LEMMA 2 *Go games are in 1-1 correspondence with simple paths starting at the empty position in the game graph.*

PROOF. The previous Lemma shows that any path corresponds to a sequence of moves, not necessarily alternating in color. Inserting a single pass before every out-of-turn move, and 2 passes at the end, produces a properly alternating and ending game. The starting node ensures that the game starts from an empty board, while simplicity of the path ensures that each move is legal. Furthermore, since any game can be stripped of its passes to produce the corresponding path, this is a bijection. •

The above Lemma applies only to rules with Positional SuperKo. With Situational SuperKo, the corresponding paths are not necessarily simple, and a position can be visited twice (once by each player).

LEMMA 3 *The game graph is strongly connected.*

PROOF. Obviously we can reach any position from the empty board, so it suffices to show that we can reach the empty position from any position. We eliminate strings one by one, without ever creating new strings. To eliminate a string, its owner repeatedly plays on its liberties while his opponent passes. Some opponent strings may get captured in the process, but ultimately, the string itself will commit suicide. •

Note that this result depends on the possibility of suicide, and fails to hold for alternative rulesets, such as the Japanese rules of Go, which forbid suicide. Under such rules, a slightly weaker property can be shown.

LEMMA 4 *On all boards except 1×1, 1×2 and 2×1, the game subgraph obtained by removing the empty position and all suicide edges is strongly connected.*

PROOF. First note that $G(1, 1)$ has no suicide-edges to remove, while $G(1, 2)$ breaks into 2 components when removing the empty position. Obviously, we can reach any position from a suitable single-stone position, so it suffices to show that we can reach every single-stone position from any position. First, one

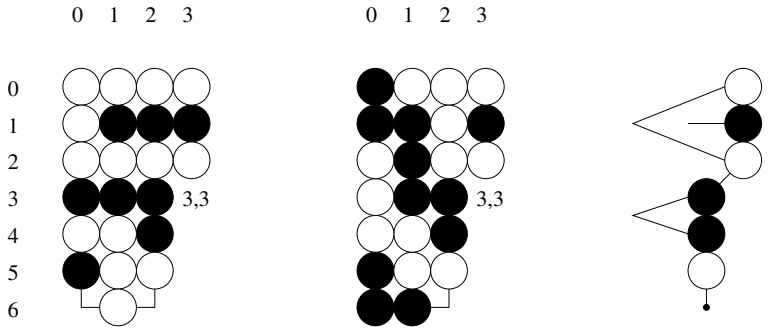


Figure 5: Two partial positions up to $(3, 3)$ and their common border state.

player, say black, repeatedly plays on a liberty of its strings. When all such plays result in suicides, the other player, white, can proceed to capture all black strings. Next, white can play until she has $mn - 1$ stones, which black then captures. Given that $mn \geq 3$, we can repeat this last phase to lead to a capture by the desired color on the desired point. •

5 Counting legal positions

The simplest way to count $L(m, n)$, the number of legal $m \times n$ positions, is by brute force, just trying all 3^{mn} positions and testing each one for legality. However, a 5×5 board already has over 400 billion possible positions, and 9×9 has over 10^{38}. Instead, we establish a correspondence between legal positions and paths in the so called *border state graph*, whose size is much more manageable. The problem thus reduces to that of counting paths of a certain length in a graph, which can be done efficiently by the method of Dynamic Programming. First we introduce the notion of partial boards, from which the border states naturally arise.

5.1 Partial Boards

Recall that we number the points $(x, y) \in \{0, \ldots, n-1\} \times \{0, \ldots, m-1\}$. We picture a go board with the point $(0, 0)$ in the top-left, x-coordinates increasing to the right, and y-coordinates increasing downward. For $0 \leq x < n$ and $0 \leq y < m$, let a partial go board up to column x and row y consist of all the points to the left of and above (x, y). It has x full columns and, if $y > 0$, one partial column of y points. Figure 5 shows two example partial $7 \times n$ positions up to $(3, 3)$.

What these positions have in common is that the set of possible completions into legal full-board positions is identical. In either case, the remainder of the position has to provide a liberty to the top white group, to the black group it

surrounds, and to the middle black group. We say that the positions share the same *border state*.

5.2 Border States

DEFINITION 2 (BORDER STATE) *A* border state*, or* state *for short, comprises the following data:*

- *the board height* m*,*
- *the size* $0 \leq y < m$ *of the partial column,*
- *the color of border points* $(x, 0), \ldots, (x, y-1), (x-1, y), \ldots, (x-1, m-1)$ *(*x *is only a symbol whose value is immaterial to the state),*
- *for each stone on the border, whether it has liberties,*
- *connections among libertyless stones.*

A state with height m *and partial column size* y *is called an* $\binom{m}{y}$*-state, or simply* y*-state if* m *is clear from context. A partial position is pseudolegal if all libertyless stones are on, or connected to, the border. A state is called* constructible *if it is the border state of some pseudolegal partial position of arbitrary width.*

Information of liberties and connections is assumed to be consistent within the border.

A partial position up to $(0, y)$ has a border state where points $(x-1, y), \ldots, (x-1, m-1)$ are off the board. We accomodate these zero-width positions by allowing the non-color 'edge' for all $(x-1, \cdot)$ points, and call the result an *edge state* instead.

In figures, libertyless stones and their connections are indicated with lines emanating to the left.

The set of constructible states is difficult to characterize, and hence to count. We therefore introduce a slightly larger class.

DEFINITION 3 *Call a* y*-state* s valid *if it satisfies all the following:*

- *connections don't cross, i.e. if 4 stones are ordered vertically as* a, b, c, d*, with* a *and* c *connected, and* b *and* d *connected, then they must all be connected.*
- *if a stone at* $(x, y-1)$ *either*
  - *has connections, but (if* $y > 1$*) not to* $(x, y-2)$*, or*
  - *has liberties, but (if* $y > 1$*)* $(x, y-2)$ *is opposite-colored,*

  then points $(x, y-1)$ *and* $(x-1, y)$ *are considered adjacent.*

LEMMA 5 *Every constructible state is valid.*

Figure 6: (a) a valid but unconstructible state (b) non-rectangular boardshape

PROOF. The non-crossing property can be seen to follow from the planarity of two-dimensional boards. If border point $(x, y-1)$ has a connection necessarily going through non-border point $(x-1, y-1)$, then the latter's neighbour $(x-1, y)$ is effectively also the former's neighbour. This virtual adjacency implies that $(x-1, y)$ must be either opposite colored, or same colored and connected to $(x, y-1)$. Similarly, if $(x, y-1)$ has a liberty necessarily obtained through $(x-1, y-1)$, then $(x, y-1)$ is effectively adjacent to $(x-1, y)$, preventing the latter from being a same colored-stone without liberties. •

The smallest example of a valid but not constructible state occurs at $m = 3$ as shown in Figure 6(a). Any partial board that connects the two white stones and provides a liberty to the black stone, will inevitably provide a liberty to the white stones as well. In other words, the fixed board height of 3 doesn't allow the white string to distance itself from the black one. If we allowed variable height board shapes such as the one shown in Figure 6(b), then the above valid state would become constructible. It can be shown that non-constructability of valid states is due entirely to the 'lack of room' to simultaneously provide liberties and connections.

We can somewhat efficiently compute the number of valid 0-border states, each of which corresponds to a balanced path of length m through the finite automaton shown later in Figure 10. In a *balanced* path the up and down connections of libertyless black and white stones match up properly like balanced parentheses of two types, e.g. '([]([]))'. Using Dynamic Programming, we can compute the number of such paths ending in a given vertex of the automaton with a given stack of pending connections. With at most $m/2$ pending connections, there are at most $2^{m/2}$ such stacks. Having all counts for paths of length l, we can in time $O(2^{m/2})$ compute all counts for paths of length $l+1$, thus taking time $O(m2^{m/2})$ overall. It took only 0.005 sec to determine the number of $\binom{19}{0}$ border states in this manner. Computing numbers of y-states for $y > 0$, numbers of edge states, and of state classes is only slightly more complicated and can be done within the same time complexity.

Table 1 shows for each m the minimum and maximum number of valid $\binom{m}{y}$-border state classes, which turn out to be achieved at $y = 0$ and $y = m-1$, respectively. The maximum is always about 45% larger than the minimum.

Only a tiny percentage of the listed valid states fails to be constructible (e.g. only 337 of the 109736 valid $\binom{9}{0}$-classes).

m	#valid $\binom{m}{0}$-classes	#valid $\binom{m}{m-1}$-classes
1	3	3
2	9	13
3	32	46
4	117	168
5	444	642
6	1712	2482
7	6742	9808
8	26973	39324
9	109736	160286
10	452863	662265
11	1894494	2772774
12	8020098	11742926
13	34320647	50258461
14	148266922	217096273
15	645949499	945567689
16	2835158927	4148642993
17	12526125303	18320946269
18	55665579032	81376671503
19	248661924718	363324268018

Table 1: Number of valid border state classes.

5.3 The border state graph

Let's see how the border state of a partial board up to (x, y) and the color of (x, y) uniquely determine the border state up to $(x, y + 1)$, as exemplified in Figure 7. If (x, y) is empty, then any libertyless stones at $(x, y-1)$ and $(x-1, y)$ as well as all stones connected to them, become stones with liberties. Otherwise, suppose (x, y) is black.

If $(x - 1, y)$ is a libertyless white stone without connections, then the new partial board is no longer pseudolegal; a case we must avoid.

If either $(x, y-1)$ or $(x-1, y)$ is empty or black with liberties then (x, y) has liberties, which it may provide to the other neighbour (if that is black without liberties).

Finally, if neither $(x, y - 1)$ nor $(x - 1, y)$ provides liberties, then (x, y) is without liberties and connects any black neighbours.

The case for (x, y) being white is identical with all colors reversed.

Similarly, the edge state of a partial board up to $(0, y)$ and the color of $(0, y)$ uniquely determine the edge state up to $(0, y+1)$ (in case $y < m-1$) or border state up to $(1, 0)$ (in case $y = m - 1$).

Definition 4 ((augmented) border state graph) *Let $B(m)$ be the directed graph whose vertices are the constructible border states of height m, and which has edges from each y-state to its 2 or 3 successor $((y + 1) \bmod m)$ states. The*

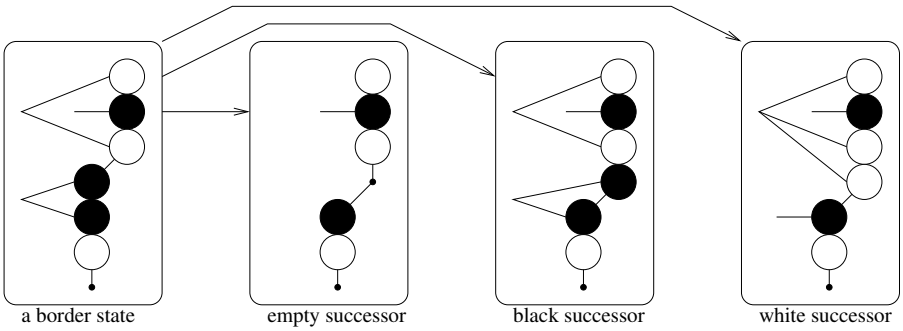


Figure 7: A border state and its 3 successors.

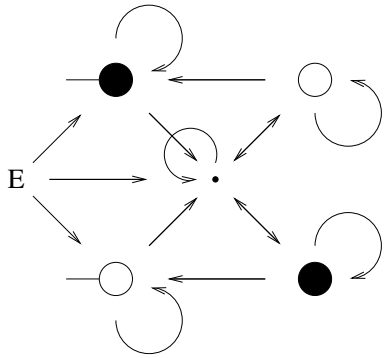


Figure 8: Augmented border state graph $AB(1)$.

augmented border state graph $AB(m)$ has additional vertices and outgoing edges for all edge states.

Figure 7 shows the 3 successors of a border state, while Figure 8 shows the augmented border state graph $AB(1)$. Border state graph $B(2)$ is too large to show in full detail, so instead of border states we show border state *classes*.

For further clarity, Figure 9 shows edges between the 9 0-state classes and 13 1-state classes separately for each direction, with the state classes ordered to minimize edge crossings. Thick edges represent the case where adding a black stone and adding a white stone lead to equivalent states in $B(2)$.

LEMMA 6 *The border state graph is strongly connected.*

PROOF. Every y-state reaches the stoneless y-state after m empty successors, and thus reaches the stoneless 0-state after another $m - y$ empty successors. From the latter, we can reach any possible column 0 state s, and hence any constructible state, as follows. Note that s can be reached in m steps from a 0-state s' that replaces each stone in s by a stone with liberties of the opposite

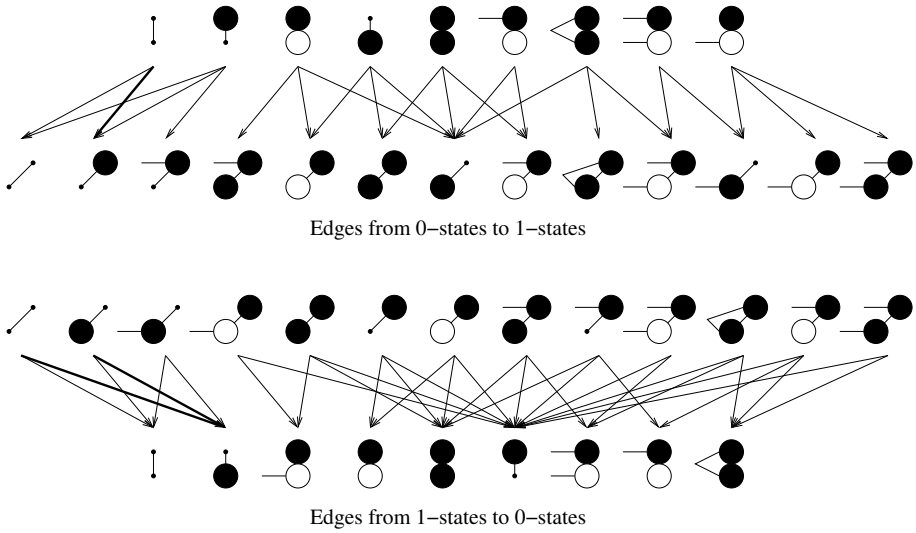


Figure 9: Edges between state classes of $B(2)$.

color, effectively forming a virtual edge. Any such state s' can clearly by reached from the stoneless 0-state in m steps. •

LEMMA 7 *There is a 1-1 correspondence between pseudolegal partial positions up to (n, y) and paths of length $mn+y$ through the augmented border state graph that start at the all-edge state.*

PROOF. By induction on $mn + y$. The unique partial position up to $(0, 0)$ corresponds to the length 0 path starting at the all-edge state. Furthermore, for a path ending at state s corresponding to a pseudolegal partial position p up to (n, y), the successors of s correspond exactly to the pseudolegal partial positions p' up to $(n, y + 1)$ (or $(n + 1, 0)$ for $y = m - 1$) that extend p by one point. •

5.4 Recurrences

DEFINITION 5 (STATE COUNTS) *For an $\binom{m}{y}$-state s, denote by $L(m, n, y, s)$ the number of pseudolegal partial positions up to (n, y) that have border/edge state s (or equivalently, the number of paths of length $mn+y$ in the augmented border state graph from the all-edge state to s). Call a y-state s* legal *if $y = 0$ and s has no libertyless stones.*

Obviously, we have

LEMMA 8 (COLOR SYMMETRY) *Let state s' be derived from state s by reversing the colors of all stones. Then $L(m, n, y, s) = L(m, n, y, s')$.*

DEFINITION 6 (STATE CLASSES) *We define a* state class*, denoted* $[s]$*, as the equivalence class of state* s *under color reversal. Call a state class* legal *when its members are, and define* $L(m,n,y,[s]) = \sum_{s' \in [s]} L(m,n,y,s')$.

Note that all equivalence classes, except for stoneless states, consist of exactly 2 states. These definitions immediately imply

LEMMA 9 $L(m,n) = \sum_{legal\ s} L(m,n,0,s) = \sum_{legal\ [s]} L(m,n,0,[s])$.

Another form of symmetry occurs in 0-states only:

LEMMA 10 (UP-DOWN SYMMETRY) *Let state* s' *be derived from* 0*-state* s *by reversing the order of points from top to bottom. Then* $L(m,n,y,s) = L(m,n,y,s')$ *and* $L(m,n,y,[s]) = L(m,n,y,[s'])$.

We refer to state classes resulting from including up-down symmetry in addition to color symmetry as up/down classes.

DEFINITION 7 (STATE COUNT VECTOR) *Denote by* $\mathbf{L}(m,n,y)$ *the state-indexed vector with elements* $L(m,n,y,s)$ *for all constructible* y*-states* s *(edge states for* $n = 0$*, border states for* $n > 0$*) and by* $\mathbf{l}_m$ *the characteristic vector of legal states of height* m.

Now Lemma 9 can be expressed as

$$L(m,n) = \mathbf{l}_m^T \mathbf{L}(m,n,0).$$

The following crucial observation forms the basis for the recurrences we derive. Since $L(m,n,y+1,s)$ equals the sum of $L(m,n,y,s')$ over all predecessor states s' of s in the augmented border state graph, it follows that the border state vectors $\mathbf{L}(m,n,y), n > 0$ are related by linear transformations $\mathbf{T}_{m,y}$, such that $\mathbf{L}(m,n,y+1) = \mathbf{T}_{m,y}\mathbf{L}(m,n,y)$ (and $\mathbf{L}(m,n+1,0) = \mathbf{T}_{m,m-1}\mathbf{L}(m,n,m-1)$). Indeed, the $\mathbf{T}_{m,y}$ appear as submatrices of the transposed adjacency matrix of the border state graph. As a consequence, successive 0-state vectors are related as

$$\mathbf{L}(m,n+1,0) = \mathbf{T}_{m,m-1}\mathbf{T}_{m,m-2}\ldots\mathbf{T}_{m,1}\mathbf{T}_{m,0}\mathbf{L}(m,n,0).$$

Thus we are led to define

DEFINITION 8 (RECURRENCE MATRIX) *Let* $\mathbf{T}_m = \mathbf{T}_{m,m-1}\ldots\mathbf{T}_{m,0}$.

This leads to a matrix power expression for $L(m,n)$:

$$L(m,n) = \mathbf{l}_m^T \mathbf{T}_m^{n-1} \mathbf{L}(m,1,0).$$

Furthermore, $L(m,n)$ can be shown to satisfy a recurrence not involving the border state counts. To simplify the following derivations, m is understood to be fixed and is dropped from the notation, so that $L(m,n) = \mathbf{l}^T\mathbf{T}^{n-1}\mathbf{L}(1,0)$.

THEOREM 1 *For fixed m, $L(m,n)$ satisfies a linear recurrence whose order is at most the number of valid 0-states.*

PROOF. Let $p(\lambda)$ be the characteristic polynomial of the $r \times r$ matrix $\mathbf{T}$,

$$p(\lambda) = \det(\lambda\mathbf{I} - \mathbf{T}) = \lambda^r + a_{r-1}\lambda^{r-1} + \cdots + a_1\lambda + a_0.$$

By the Cayley-Hamilton theorem, $p(\mathbf{T}) = \mathbf{0}$. It follows that

$$\mathbf{T}^r = -(a_{r-1}\mathbf{T}^{r-1} + \cdots + a_1\mathbf{T} + a_0\mathbf{I})$$

and by multiplication by $\mathbf{T}^{k-1}$ that

$$\mathbf{T}^{r+k-1} = -(a_{r-1}\mathbf{T}^{r+k-2} + \cdots + a_1\mathbf{T}^k + a_0\mathbf{T}^{k-1})$$

for all $k \geq 1$. Multiplying by $\mathbf{l}^T$ on the left and $\mathbf{L}(1)$ on the right yields

$$\begin{aligned} L(m,k+r) &= \mathbf{l}^T\mathbf{T}^{r+k-1}\mathbf{L}(1,0) \\ &= -(a_{r-1}L(m,k+r-1) + \cdots + a_1L(m,k+1) + a_0L(m,k)) \end{aligned}$$

for all $k \geq 1$, proving the theorem. •

By expressing $\mathbf{T}$ in terms of state classes rather than states, and modifying $\mathbf{l}^T$ and $\mathbf{L}(1,0)$ accordingly (as illustrated in section 5.4.1), we obtain a stronger upperbound on the order, namely the number of valid 0-state classes as given in Table 1. Finally, that bound may be nearly halved again by using up/down classes.

The structure of solutions to linear recurrences is well known [1]. Theorem 1 implies

COROLLARY 3 *For fixed m, $L(m,n)$ can be written in the form*

$$L(m,n) = \sum_k q_k(n) l_k^n,$$

where l_k are the distinct eigenvalues of $\mathbf{T}$, and q_k are polynomials of degree at most multiplicity$(l_k) - 1$.

Notice that some of the terms may vanish but not the largest eigenvalue, which we have additional information about. First a technical lemma is needed.

LEMMA 11 *If $l_k \neq 0$ is an eigenvalue of $\mathbf{T}$ of multiplicity 1 with corresponding left and right eigenvectors $\mathbf{e}^T$ and $\mathbf{f}$ then q_k is the constant $\frac{\mathbf{l}^T\mathbf{f}\mathbf{e}^T\mathbf{L}(1,0)}{l_k}$.*

PROOF. Let $\mathbf{T} = \mathbf{V}\mathbf{J}\mathbf{V}^{-1}$ be the Jordan canonical decomposition of $\mathbf{T}$. Thus $L(m,n) = \mathbf{l}^T\mathbf{T}^{n-1}\mathbf{L}(1,0) = \mathbf{l}^T\mathbf{V}\mathbf{J}^{n-1}\mathbf{V}^{-1}\mathbf{L}(1,0)$. Since l_k has multiplicity one, it must have a single corresponding Jordan block of size 1. Due to the structure of J only this block can provide l_k^n terms to $L(m,n)$, more precisely $\mathbf{l}^T e l_k^{n-1} f^T \mathbf{L}(1,0)$ •

THEOREM 2 *There exist* $a_m > 0$, $0 < \lambda_m \leq 3^m$, *and* $0 < \phi_m < 1$ *such that*

$$L(m,n) = a_m \lambda_m^n (1 + r(m,n))$$

with $r(m,n) = O(\phi_m^n)$.

PROOF. By lemma 6 it follows that $\mathbf{T}$ is regular, i.e. $\mathbf{T}^k$ is (elementwise) positive for some k. Since, by construction, $\mathbf{T}$ is also non-negative, the Perron-Frobenius theorem guarantees the existence of a real positive eigenvalue l_1 with the properties that it has multiplicity one, is larger in magnitude than all other eigenvalues, and has left and right eigenvectors $\mathbf{e}^T$ and $\mathbf{f}$ with all elements positive. From Lemma 11 it follows that $q_1(n) = \frac{\mathbf{l}^T \mathbf{f} \mathbf{e}^T \mathbf{L}(1,0)}{l_k}$, which is guaranteed to be a positive constant since $\mathbf{l}^T$ and $\mathbf{L}(1,0)$ are non-negative and not all zero whereas $\mathbf{e}$ and $\mathbf{f}$ are positive. Now $L(m,n) = q_1 l_1^n (1 + \sum_{k>1} \frac{q_k(n)}{q_1} \frac{l_k}{l_1}^n)$ and the claimed result follows (for any $\max_{k>1} \frac{|l_k|}{l_1} < \phi_m < 1$). The upper bound on λ_m follows necessarily from the trivial upper bound $L(m,n) \leq 3^{mn}$. •

The recursion coefficients are most easily obtained by computing $L(m,n)$ for $n = 1, \ldots, 2s$ by the dynamic programming algorithm in section 5.5. Since we know that the sequence must satisfy *some* linear recurrence, the problem is reduced to determining the minimal order and the corresponding coefficients. Moreover, we know that the minimal order is upper-bounded by the number of valid state classes s given in Table 1.

LEMMA 12 *Assume that the sequence* $x(1), x(2), \ldots$ *satisfies the linear recurrence*

$$x(k+r) = c_{r-1}x(k+r-1) + \cdots + c_1 x(k+1) + c_0 x(k).$$

Then the coefficients c_i *satisfy the equation system*

$$\begin{pmatrix} x(1) & x(2) & \ldots & x(r) \\ x(2) & x(3) & \ldots & x(r+1) \\ \vdots & \vdots & \ddots & \vdots \\ x(r) & x(r+1) & \ldots & x(2r-1) \end{pmatrix} \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{r-1} \end{pmatrix} = \begin{pmatrix} x(r+1) \\ x(r+2) \\ \vdots \\ x(2r) \end{pmatrix}.$$

Furthermore, r *is the minimal order of recurrence iff the above matrix is non-singular, in which case the coefficients are uniquely determined.*

As we do not know the minimal order of recurrence, we form matrices for increasing r. For each non-singular one, we compute the recurrence coefficients and verify the recurrence for all s elements $L(m, s+1)$ through $L(m, 2s)$. The smallest verifiable r is the minimal order. For efficient computations, the Berlekamp-Massey algorithm [11] can be used together with modular arithmetic and the Chinese Remainder Theorem.

5.4.1 $1 \times n$ Boards

For one-dimensional boards, with $m = 1$, Figure 8 shows the five possible border states "empty", "black with liberty", "white with liberty", "black without liberty", and "white without liberty". The first three are legal, so $\mathbf{l} = (1, 1, 1, 0, 0)^T$. The state count transformation is given by the transposed adjacency matrix

$$\mathbf{T} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}$$

and the initial state count with one column gives $\mathbf{L}(1, 0) = (1, 0, 0, 1, 1)^T$. In this case $\mathbf{T}$ is invertible and we can write $\mathbf{L}(0, 0) = \mathbf{T}^{-1}\mathbf{L}(1, 0) = (-1, 1, 1, 0, 0)^T$ and

$$L(1, n) = \mathbf{l}^T\mathbf{T}^{n-1}\mathbf{L}(1, 0) = \mathbf{l}^T\mathbf{T}^n\mathbf{L}(0, 0) =$$
$$\begin{pmatrix} 1 & 1 & 1 & 0 & 0 \end{pmatrix} \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 \end{pmatrix}^n \begin{pmatrix} -1 \\ 1 \\ 1 \\ 0 \\ 0 \end{pmatrix},$$

which gives the sequence $1, 5, 15, 41, 113, 313, 867, 2401, 6649, 18413, \ldots$

The characteristic polynomial of $\mathbf{T}$ is $p(\lambda) = \det(\lambda\mathbf{I} - \mathbf{T}) = \lambda^5 - 5\lambda^4 + 8\lambda^3 - 6\lambda^2 + 3\lambda - 1$. It follows that $L(1, n)$ satisfies the recurrence

$$L(1, k + 5) = 5L(1, k + 4) - 8L(1, k + 3) + 6L(1, k + 2) - 3L(1, k + 1) + L(1, k).$$

This is not a minimal order recurrence, however. Using state classes instead (empty, stone with liberty, or stone without liberty) yields

$$\mathbf{l} = \begin{pmatrix} 1 & 1 & 0 \end{pmatrix}^T,$$
$$\mathbf{T} = \begin{pmatrix} 1 & 1 & 1 \\ 2 & 1 & 0 \\ 0 & 1 & 1 \end{pmatrix},$$
$$\mathbf{L}(1, 0) = \begin{pmatrix} 1 & 0 & 2 \end{pmatrix}^T$$

and a characteristic polynomial $p(\lambda) = \lambda^3 - 3\lambda^2 + \lambda - 1$, leading to the minimal recurrence

$$\boxed{L(1, k + 3) = 3L(1, k + 2) - L(1, k + 1) + L(1, k).}$$

These coefficients can also be computed directly from $L(1, 1), \ldots, L(1, 6)$ according to Lemma 12 as

$$\begin{pmatrix} 1 & 5 & 15 \\ 5 & 15 & 41 \\ 15 & 41 & 113 \end{pmatrix}^{-1} \begin{pmatrix} 41 \\ 113 \\ 313 \end{pmatrix} = \begin{pmatrix} 1 \\ -1 \\ 3 \end{pmatrix}.$$

可以考虑搞一个生成函数的精确形式

$L(1,n)$ can be written in the form of Corollary 3 as

$$L(1,n) \sim 0.694 \cdot 2.769^n + (0.153 - 0.812i) \cdot (0.115 + 0.590i)^n + \\ + (0.153 + 0.812i) \cdot (0.115 - 0.590i)^n,$$

where the constants involved are solutions to cubic equations. In particular the largest eigenvalue can be written in closed form as

$$\lambda_1 = 1 + \frac{1}{3}\left((27 + 3\sqrt{57})^{\frac{1}{3}} + (27 - 3\sqrt{57})^{\frac{1}{3}}\right) \sim 2.769292354.$$

5.4.2 $2 \times n$ up to $9 \times n$ Boards

For $2 \times n$ boards, Table 1 shows 9 state classes, but up-down symmetry reduces this to 7 up/down classes, matching the recurrence order:

$$L(2,n+7) = 10L(2,n+6) - 16L(2,n+5) + 31L(2,n+4) - 13L(2,n+3) + \\ + 20L(2,n+2) + 2L(2,n+1) - L(2,n).$$

This sequence starts $1, 5, 57, 489, 4125, 35117, 299681, \ldots$.

For $3 \times n$ the number of constructible up/down classes is 21, but the minimal recurrence order is only 19:

$$\begin{aligned} L(3,n+19) = {} & 33L(3,n+18) - 233L(3,n+17) + 1171L(3,n+16) - \\ & - 3750L(3,n+15) + 9426L(3,n+14) - 16646L(3,n+13) + \\ & + 22072L(3,n+12) - 19993L(3,n+11) + 9083L(3,n+10) + \\ & + 1766L(3,n+9) - 4020L(3,n+8) + 6018L(3,n+7) - \\ & - 2490L(3,n+6) - 5352L(3,n+5) + 1014L(3,n+4) - \\ & - 1402L(3,n+3) + 100L(3,n+2) + 73L(3,n+1) - 5L(3,n). \end{aligned}$$

The recurrences for $m \geq 4$ are too long to show here. The number of constructible up/down classes, minimum recurrence order, and the λ_m value from theorem 2 are listed in Table 2 for $1 \leq m \leq 9$.

5.4.3 Legal Probabilities and Markov Chains

Dividing the number of legal positions $L(m,n)$ by the total number of positions 3^{mn} gives the probability that a random position on an $m \times n$ board is legal. With this interpretation, it is natural to consider the distribution of border states as a Markov Chain. First, however, it is necessary to extend the border state space with "illegal" states (one for each partial column size) which represent partial boards that aren't pseudolegal, and make all states have outdegree 3. With this modification, the vectors $\mathbf{p}(m,n) = \frac{1}{3^{mn}}\mathbf{L}(m,n,0)$ are probability distributions over the border states and $\mathbf{P}_m = \frac{1}{3^m}\mathbf{T}_m$ is a probability transition matrix, so that $\mathbf{p}(m,n+1) = \mathbf{P}_m\mathbf{p}(m,n)$ describes the change in distribution of border states when a new random column is added. Since $\mathbf{P}_m$ is a stochastic matrix, i.e. the sum of all columns is one, it has a largest eigenvalue of 1.

size	classes	order	a_m	$\sqrt[m]{\lambda_m}$
$1 \times n$	3	3	0.69412340909080771809	2.7692923542386314
$2 \times n$	7	7	0.77605920648443217564	2.9212416045359486
$3 \times n$	21	19	0.76692462372625158688	2.9412655443486972
$4 \times n$	64	57	0.73972591465609392167	2.9497646496768897
$5 \times n$	242	217	0.71384057986002504205	2.9549337288382067
$6 \times n$	880	791	0.68921150040083474629	2.9583903342140907
$7 \times n$	3453	3107	0.66545979340188479816	2.9608618349040166
$8 \times n$	13556	12110	0.64252516474515096185	2.9627168070252408
$9 \times n$	55193	49361	0.62038058380200867949	2.9641603664723

Table 2: Small board recurrences.

The corresponding eigenvector has a 1 for the "illegal" state and zero for all others, i.e. the illegal state is an absorbing state, which is not surprising since pseudolegality can only get lost under board extension, not regained.

5.5 The Dynamic Programming algorithm

The algorithm starts from the unit state vector $\mathbf{L}(m, 0, 0)$ (which has a 1 for the all-edge state only), and then performs mn linear transformations (the first m of which operate on edge states, while the rest, of the form $\mathbf{T}_{m,y}$, operate on border states) to obtain $\mathbf{L}(m, n, 0)$. Instead of keeping exact counts $L(m, n, y, s)$ as vector elements, we use state class counts modulo some number M close to 2^{64}. Running the algorithm $\lceil mn \log_2(3)/64 \rceil$ times with different, relatively prime, moduli gives us a set of equations

$$L(m, n) = a_i \bmod M_i,$$

which is readily solved using the Chinese Remainder Theorem (CRT). This technique trades off memory and diskspace (which are more constrained) for time. There is an interesting side-benefit of using chinese remaindering: automatic error detection. If, for instance, a single bitflip in memory should corrupt one of the remainders, then reconstruction of the full count from the remainders will yield a value that's way off, which will be obvious when comparing with the approximation formula we'll derive in a later section.

The heart of the algorithm is an efficient representation of border state classes, using just 3 bits per point, or $3m$ bits for a state class. This makes the standard height of $m = 19$ fit comfortably in 64-bit integers. The non-crossing connections can be represented with just 2 booleans per libertyless stone: whether it has a connection above it, and whether it has a connection below it. The representation further exploits the fact that neighbouring points in the border highly constrain each other. Figure 10 shows possible transitions from one point to the next in a "bump-free" 0-state. Upward and downward pointing arrows from the line indicating lack of liberties represent the two

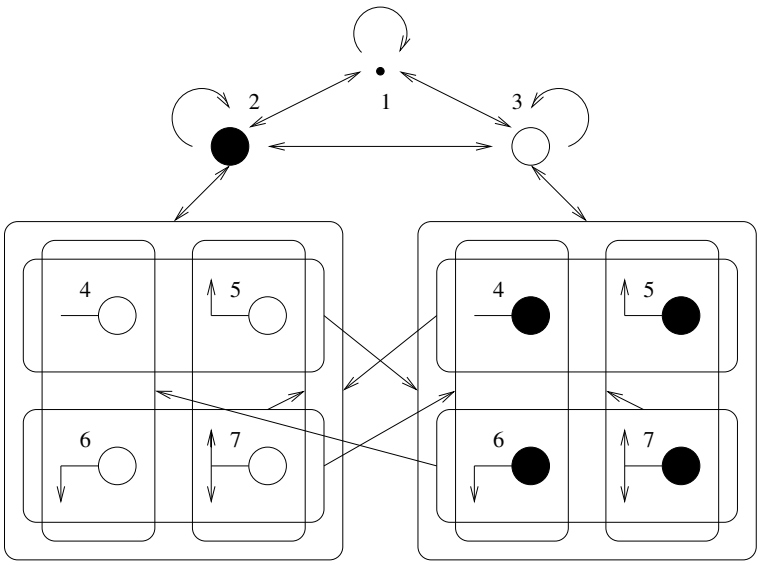


Figure 10: Intra-border transitions.

boolean flags.

Edges between boxed sets of points indicate the presence of edges from all points in one set to all points in the other. Next to each point is shown its 3-bit code. Note that no point has two different transitions to same numbered points. This reflects the fact that two libertyless adjacent stones have the same color if and only if they are connected, and a stone with liberties cannot be adjacent to a libertyless stone of the same color. The algorithm uses code 0 for 'edge' points in edge states. Two pieces of information are still lacking; the color of a libertyless stone at $(x, 0)$, and the color of a libertyless stone on $(x-1, y)$, which is not adjacent to the previous border point at $(x, y-1)$. However, since we represent state classes rather than states, we can assume that $(x-1, y)$, if non-empty, is always white. In this case, the color of a libertyless stone at $(x, 0)$ can be stored in the boolean indicating connections above, since the latter is always false. If $(x-1, y)$ is empty, then we can assume any stone on $(x, 0)$ is white. If both $(x-1, y)$ and $(x, 0)$ are empty then we can normalize the color of e.g. the bottom-most stone on the border.

Note that this is the lowest of 3 levels of graphs considered in this paper: a game of Go is a path in the game graph. whose nodes are paths in the border state graph, whose nodes are paths in the 11-node intra-border graph[2].

In order to compute $\mathbf{L}(m, n, y+1) = \mathbf{T}_{m,y}\mathbf{L}(m, n, y)$, the algorithm loops over all state-count pairs (s, i) in $\mathbf{L}(m, n, y)$, computes the 2 or 3 successors s' of

[2]Ignoring such details as edge states/points and the cross-column transition for non-0 states.

s, and stores the new pairs (s', i) in some data structure. Matching pairs (s', i) and (s', j) need to be combined into a single pair $(s', i + j \bmod M)$, which is easy if all states can be kept in memory. For large computations, like $m = 19$, this is not possible and the state-count pairs need to be stored on disk. To allow for efficient combining of states, we collect as many pairs as possible in memory, before flushing them all to disk in sorted order. To save space, we store only the differences between consecutive states (using 7 bits per byte, with the 8th bit indicating the last byte). Note that a state can appear in as many files as it has predecessors in $\mathbf{L}(m, n, y)$. The total number of state-count-pairs in all output files is thus larger than the number of $(y+1)$-states by a factor in the range $[1, 3]$. This *redundancy* is the average number of files in which a $(y + 1)$-state occurs. Merging all output files while combining like pairs removes this redundancy and produces the required $\mathbf{L}(m, n, y + 1)$. To keep redundancy small, we want the different predecessors of a state to be close together in the input ordering, so that combination can take place before a memory flush. This ordering depends on how the m 3-bit fields are joined into a $3m$ bit integer. We do this in different ways, such that the most variable fields, namely, those close to y, end in up the less significant bits of the $3m$ bit integer.

It turns out that the computation of $\mathbf{T}_{m,y}$ as described above, *precisely* fits the Google *MapReduce* framework described in [13]. Our map function maps a state-count pair (s, i) to a list of 2 or 3 new pairs (s', i), while our reduce function merely sums modulo M.

Having completed all mn linear transformations, which gives us $\mathbf{L}(m, n, 0)$, we then sum all counters of legal states to get the desired $L(m, n) \bmod M$ result.

5.6 Complexity

The main factor in both time and space complexity is the number of state classes s. This may be upperbounded by ignoring the connection constraints (equivalent to balancing parentheses) and computing the largest eigenvalue of the intra-border transition matrix, which turns out to be $\lambda \sim 5.372$ (the largest root of $(\lambda - 2)(\lambda^2 - 5\lambda - 2)$). The number of paths of length m through the intra-border transition graph, and hence s, is therefore bounded by $O(\lambda^m)$. We thus need at most $\lambda^m(3m + 64)$ bits plus the overhead for the datastructure which is at most linear, for a space complexity of $O(m\lambda^m)$.

这里算法的复杂度可以可以用 symbolic method 再仔细分析一下，可能会得到更紧的结果

For time complexity, we have the product of the number of moduli, which is $\lceil mn \log_2(3)/64 \rceil$, the number mn of passes, the number $O(\lambda^m)$ of states, and the amount of work $O(m)$ per state, which results in time $O(m^3 n^2 \lambda^m)$.

5.7 Results

Table 5.7 shows the number of legal positions.

The $L(17, 17)$ computation took over 8000 CPU-hours and 3TB of disk space generously provided by the Opteron based Linux Cluster of the INS group at the Center for Mathematics and Computer Science (CWI) in Amsterdam. Paul Leyland used the General Number Field to find 3 prime factors including

n	#digits	$L(n,n)$
1	1	1
2	2	57
3	5	12675
4	8	24318165
5	12	414295148741
6	17	62567386502084877
7	23	83677847847984287628595
8	30	990966953618170260281935463385
9	39	103919148791293834318983090438798793469
10	47	96498428501909654589630887978835098088148177857
11	57	79347486681658226682093667179018913232167338311218515189 9
12	68	57774258489513238998237970307483999327287210756991189655942651331169
13	80	37249792307686396442294904767024517674249157948208717533254799550970595875237705
14	93	21266773290036622424978935765044059809880586108326912719662387221322819635245544757502970132 5
15	107	10751464308361383118768413754866123809733788820327844402764601662870883601711298309339239868998337801509491
16	121	48130669638227554164290560224842996464868741009672492639447195999756074598505022220395911493314318055246554674530670423 77
17	137	19079388919628199204605726181850465220151058338147922243967269231944059187214767997105992341735209230667288462179090073659712583262087437
18	153	669723114288829212892740188841706543509937780640178732810318337696945624428547218105214326012774371397184848890970111836283470468812827907149926502347633
19	171	208168199381979984699478633344862770286522453884530548425639456820927419612738015378525648451698519643907259916015628128546089888314427129715319317557736620397247064840935

2*10^170

Table 3: Number of legal $n \times n$ positions.

n	$L(n,n)$
18	$\sim 0.0173 \cdot 3^{324} \sim 6.6 \cdot 10^{152}$
19	$\sim 0.01196 \cdot 3^{361} \sim 2.082 \cdot 10^{170}$
29	$\sim 1.2 \cdot 10^{-4} \cdot 3^{841} \sim 2.2 \cdot 10^{397}$
39	$\sim 2.4 \cdot 10^{-7} \cdot 3^{1521} \sim 1.2 \cdot 10^{719}$
49	$\sim 9 \cdot 10^{-11} \cdot 3^{2401} \sim 3.5 \cdot 10^{1135}$
59	$\sim 7 \cdot 10^{-15} \cdot 3^{3481} \sim 5 \cdot 10^{1646}$
69	$\sim 1.1 \cdot 10^{-19} \cdot 3^{4761} \sim 4 \cdot 10^{2252}$
79	$\sim 3 \cdot 10^{-25} \cdot 3^{6241} \sim 2 \cdot 10^{2953}$
89	$\sim 2 \cdot 10^{-31} \cdot 3^{7921} \sim 4 \cdot 10^{3748}$
99	$\sim 2 \cdot 10^{-38} \cdot 3^{9801} \sim 4 \cdot 10^{4638}$

Table 4: Estimated Number of legal $n \times n$ positions.

46542825577 and 25180265792350457825046049349071615895 29. The $L(18,18)$ computation took over 50000 CPU-hours and 4PB of disk IO, generously provided by the Intel x86 Linux Cluster of the IAS School of Natural Sciences in Princeton. The smaller of two prime factors found with Dario Alejandro Alpern's ECM implementation is 7176527950749135946361. The $L(19,19)$ computation took over 250000 CPU-hours and 30PB of disk IO, generously provided by the Intel x86 Linux clusters at the IAS School of Natural Sciences in Princeton, the IDA Center for Communications Research, also in Princeton, and on a HP Helion Cloud server. The first 7 of 8 prime factors are 5, 401, 4821637, 964261621, 2824211368611548437, 219846696500237600175961330792275 7, and 659486468368075679414404343174041 97.

5.8 Heuristic Sampling Results

Pang Chen [9], improving on earlier work of Knuth and Purdom, developed an elegant unbiased estimator of treesize that takes advantage of heuristic knowledge in the form of a stratifier. This is a function on tree nodes which must decrease when following tree edges, and which should correlate with subtree size, since this determines the estimator variance. Pang demonstrated the power of his Heuristic Sampling algorithm on the problem of counting uncrossed knight's tours on a chess board. As stratifier he used the number of unvisited squares reachable by *currently valid* knight moves.

We applied his technique to the problem of counting legal positions by estimating the size of the tree of successive border states to depth n^2. As stratifier we use the pair (remaining depth, m minus the number of libertyless strings). For each boardsize of interest, we took the average of between 10^5 and 10^7 runs of the Heuristic Sampling algorithm, with the results shown in Table 4.

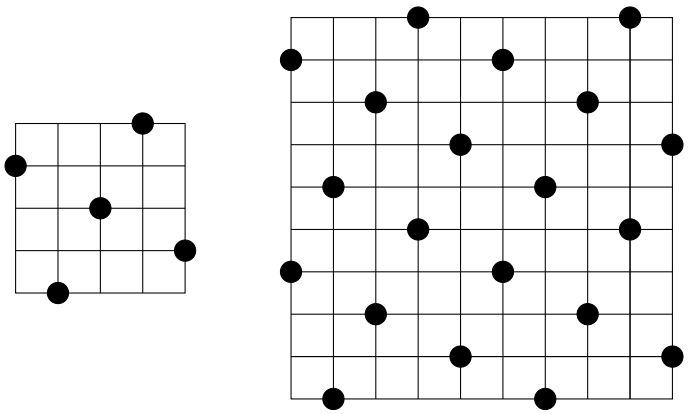


Figure 11: Knight subset of points on 5×5 and 10×10 boards.

5.9 Asymptotic Bounds

Let K denote the set of points on an $m \times n$ board, reachable from one (3,3) point with 'orthogonal' knight moves, as shown in Figure 11. For simplicity, assume that m and n are divisible by 5, so that set K has size $mn/5$ (for other m, n, $|K|$ equals $mn/5$ rounded up or down). We use K to derive both lower and upper bounds on $L(m, n)$.

THEOREM 3 *For m, n divisible by 5,*

$$3^{\frac{4mn}{5}}(1-\frac{2}{81})^{\frac{2(m+n)}{5}} \leq L(m,n) \leq 3^{mn}(1-\frac{2}{81})^{\frac{2(m+n)}{5}}(1-\frac{2}{243})^{\frac{mn-2(m+n)}{5}}.$$

PROOF. For the lower bound, color the points in K empty, and all other points randomly. Then illegality can only arise at the $2(m+n)/5$ points on the edge that neither belong to K nor neighbour K. For each such point, the probability of being a libertyless stone is $2 \cdot 3^{-4} = 2/81$, and these events are independent, so the whole position is legal with probability $(1-2/81)^{2(m+n)/5}$.

For the upper bound, we color all points randomly and only check if any point in K is a one-stone string without liberties. For each of the $2(m+n)/5$ edge points of K this happens with probability $2/81$ and for each of the $mn/5 - 2(m+n)/5$ interior points of K this happens with probability $2 \cdot 3^{-5} = 2/243$. Again, these events are all independent. •

5.10 The base of liberties

The previous section shows that $L(m, n)^{1/mn}$ is roughly between $3^{4/5} \sim 2.4$ and $3(1-\frac{2}{243})^{1/5} \sim 2.995$. In this section we prove that $L(m, n)^{1/mn}$ in fact converges to a specific value L, which we call the *base of liberties*. This is the

2-dimensional analogue of the 1-dimensional growth rate $\lambda_1 \sim 2.7693$ derived in Section 5.4.1.

Fix m and n. Consider any $M = q_m m + r_m$ and $N = q_n n + r_n$ with $0 \le r_m < m, 0 \le r_n < n$. Since tiling legal positions together preserves legality, we have

$$L(M,N) \ge L(m,n)^{q_m q_n} L(m,r_n)^{q_m} L(r_m,n)^{q_n} L(r_m,r_n).$$

This proves

THEOREM 4 *$\ln L(m,n)$ is superadditive in both arguments.*

THEOREM 5 *$\lim_{\min(m,n)\to\infty} L(m,n)^{1/mn}$ converges to some value L.*

PROOF. We extend the proof of Fekete's theorem to 2 dimensions:

证明了 base of liberty 的存在性

$$\liminf_{M,N\to\infty} \frac{\ln L(M,N)}{MN} \ge \liminf_{M,N\to\infty} \frac{q_m q_n}{MN} \ln L(m,n) + \frac{q_m}{MN} \ln L(m,r_n) +$$

$$+ \frac{q_n}{MN} \ln L(r_m,n) + \frac{1}{MN} \ln L(r_m,r_n) = \frac{\ln L(m,n)}{mn}.$$

Since m and n were arbitrary, we get

$$\liminf_{M,N\to\infty} \frac{\ln L(M,N)}{MN} \ge \sup_{m,n\ge 1} \frac{\ln L(m,n)}{mn},$$

hence $\frac{\ln L(m,n)}{mn}$ converges to some value $\ln L$. •

5.11 An Asymptotic Formula

Theorems 5 and 2 together imply that $a_m^{1/mn} \lambda_m^{1/m}$, and hence $\lambda_m^{1/m}$ (since n can go to infinity arbitrarily faster than m), converge to L. Table 2 confirms that the λ_m values behave roughly as L^m for some L. In this section we extend Theorem 2 to derive a much stronger result, albeit contingent on a conjecture about how fast the subdominant terms disappear. The central idea is that since $\ln(L(m,n))$ is asymptotically linear in n for each m, and symmetric, it can be expected to be asymptotically bilinear in m and n.

The parameters guaranteed by Theorem 2 may be obtained from $L()$ as follows:

COROLLARY 4

$$\lambda_m = \lim_{n\to\infty} \frac{L(m,n+1)}{L(m,n)},$$

$$a_m = \lim_{n\to\infty} \frac{L(m,n)}{\lambda_m^n},$$

$$r(m,n) = \frac{L(m,n)}{a_m \lambda_m^n} - 1.$$

In practice the ϕ_m values are small enough that a number of a_m, λ_m, and $r(m,n)$ can be computed with good accuracy from the $L(m,n)$ values that are practical to compute by the dynamic programming algorithm. Before looking further at these we derive a number of lemmas about symmetric and asymptotically linear functions. Recall that the discrete derivative Δ is defined by $\Delta f(n) = f(n+1) - f(n)$.

First a discrete integration result needed in Lemma 15;

LEMMA 13 *If $\Delta g(n) = O(\phi^n)$ for some $0 \le \phi < 1$ then $g(n) = \gamma + O(\phi^n)$ for some γ. Likewise $\Delta g(n) = O(n\phi^n)$ implies $g(n) = \gamma + O(n\phi^n)$ for some γ.*

PROOF. Since $\sum_{i \ge n} \Delta g(i) = O(\sum_{i \ge n} \phi^i) = O(\phi^n)$, $\lim_{n->\infty} g(n)$ converges to some value γ, and the conclusion follows. The other case is similar, with convergence implied by $\sum_{i \ge n} i\phi^i = \frac{n\phi^n}{1-\phi} + \frac{\phi^{n+1}}{(1-\phi)^2} = O(n\phi^n)$. •

LEMMA 14 *Let $x(m,n) = u(m) + nv(m) + w(m,n)$ be symmetric. Then*

$$\Delta^2 v(n) = v(n+2) - 2v(n+1) + v(n) = w(n+1,n+2) - w(n+2,n+1)$$
$$+w(n,n+1) - w(n+1,n) - w(n,n+2) + w(n+2,n)$$

and

$$\Delta u(n) = -nv(n+1) + (n+1)v(n) - w(n+1,n) + w(n,n+1).$$

PROOF.

$$\begin{aligned}
0 &= x(n+2,n+1) - x(n+1,n+2) \\
&\quad + x(n+1,n) - x(n,n+1) + x(n,n+2) - x(n+2,n) \\
&= u(n+2) + (n+1)v(n+2) + w(n+2,n+1) \\
&\quad - u(n+1) - (n+2)v(n+1) - w(n+1,n+2) \\
&\quad + u(n+1) + nv(n+1) + w(n+1,n) - u(n) - (n+1)v(n) - w(n,n+1) \\
&\quad + u(n) + (n+2)v(n) + w(n,n+2) - u(n+2) - nv(n+2) - w(n+2,n) \\
&= v(n+2) - 2v(n+1) + v(n) - w(n+1,n+2) + w(n+2,n+1) \\
&\quad - w(n,n+1) + w(n+1,n) + w(n,n+2) - w(n+2,n),
\end{aligned}$$

$$\begin{aligned}
0 &= x(n+1,n) - x(n,n+1) \\
&= u(n+1) + nv(n+1) + w(n+1,n) - u(n) - (n+1)v(n) - w(n,n+1).
\end{aligned}$$

•

LEMMA 15 *If additionally all $w()$ terms in the above are $O(\phi^n)$ for some $\phi < 1$, then there exist α, β and λ such that*

$$v(m) = \beta + \lambda m + O(\phi^m),$$
$$u(m) = \alpha + \beta m + O(m\phi^m).$$

PROOF. Lemma 14 together with the assumption give $\Delta^2 v(n) = O(\phi^n)$. Applying Lemma 13 once to $g(n) = \Delta v(n)$ gives $\Delta v(n) = \lambda + O(\phi^n)$ and applying it again to $g(n) = v(n) - \lambda n$ gives $v(n) = \beta + \lambda n + O(\phi^n)$.

Lemma 14 also gives

$$\begin{aligned}\Delta u(n) &= -nv(n+1) + (n+1)v(n) - w(n+1,n) + w(n,n+1) \\ &= -n(\beta + \lambda(n+1) + O(\phi^{n+1})) + (n+1)(\beta + \lambda n + O(\phi^n)) + O(\phi^n) \\ &= \beta + O(n\phi^n).\end{aligned}$$

A third application of Lemma 13 to $g(n) = u(n) - \beta n$ gives $u(n) = \alpha + \beta n + O(n\phi^n)$. •

COROLLARY 5 *Under the assumptions above, for $n = \Theta(m)$,*

$$x(m,n) = \alpha + \beta(m+n) + \lambda mn + O(m\phi^m).$$

Taking natural logarithms of the expression for $L(m,n)$ in Theorem 2 we have

$$\ln(L(m,n)) = \ln(a_m) + n\ln(\lambda_m) + \ln(1 + r(m,n)),$$

which is in the expected form with $x(m,n) = \ln(L(m,n))$, $u(m) = \ln(a_m)$, $v(m) = \ln(\lambda_m)$, and $w(m,n) = \ln(1 + r(m,n))$. The question is whether $\ln(1 + r(m,n))$ satisfies the conditions on $w()$ in the lemmas.

Although one can show that for fixed m, $\ln(L(m,n)) - \alpha - \beta(m+n) - \lambda mn = \Omega(n)$, we do expect this quantity to vanish for proportional n:

CONJECTURE 1 *$r(m,n) = O(\phi^m)$ for some $\phi < 1$ and $n = \Theta(m)$.*

(This should be read as: for any constant $c \geq 1$, there exists a $\phi < 1$ such that $r(m,n) = O(\phi^m)$ for all $m/c \leq n \leq cm$.) This assumption suffices to apply Corollary 5 to $\ln L(m, \Theta(m))$, which, after exponentiation, gives

THEOREM 6 *Conjecture 1 implies*

$$L(m,n) = A\ B^{m+n} L^{mn} (1 + O(m\phi^m))$$

for some constants A, B, $\phi < 1$, and $n = \Theta(m)$.

这个猜想估计比较难证

是一个可以深入研究的方向

The constants A, B, and L can all be computed as limits of expressions involving legal counts of square and almost-square boards.

COROLLARY 6 (CONTINGENT ON CONJECTURE 1)

$$\begin{aligned}L &= \lim_{n\to\infty} \frac{L(n,n)L(n+1,n+1)}{L(n,n+1)^2}, \\ B &= \lim_{n\to\infty} \frac{L(n,n+1)}{L(n,n)L^n} = \lim_{n\to\infty} \frac{L(n,n)}{L(n,n-1)L^n}, \\ A &= \lim_{n\to\infty} \frac{L(n,n)}{B^{2n}L^{n^2}}.\end{aligned}$$

n	$L(n, n+1)$
1	5
2	489
3	321689
4	1840058693
5	93332304864173
6	41945191530093646965
7	16693129760966791272789852 1
8	588274886643237065567437275212 3193
9	183573861389984542114026236485364470689 1109
10	507258858864732765845786251821669685488516949098 7149
11	124118554774307129694783556890846966815009879092863579679259393
12	268925540588602721169725623664159201380070959805515589080009823324057 43333
13	515959556656856811665975668668051814355963395026956992938234222736569704773734152003 73
14	876571894704740435776253865561651594678577903166188258472955681125289495868953613359454403019145877
15	131870512244645759297888472329947870580266256924485681728458086578687538959472921550847035733890182662513180743513
16	175669398745227675074920433327787366374551886098434473836510647159450821039563378374569811240252763776406712988379278644250456677
17	207220542761902330303958752023639012175427407271878460943399819693328260806703631440346520296370029734115221628675057659362745939297939748796407 7
18	216450089279078275314395453480468424469694873576469893709517750563261490751122922463339745178577954008324586419548071995019779454584564790800309660950831580481393
19	200203194086297695671447973013557850996986259152430382611235007734890620740154339541587081797890280045754305529783678738457045887237708512899422163924031484980226164357409684272 61

Table 5: Legal counts of almost square boards.

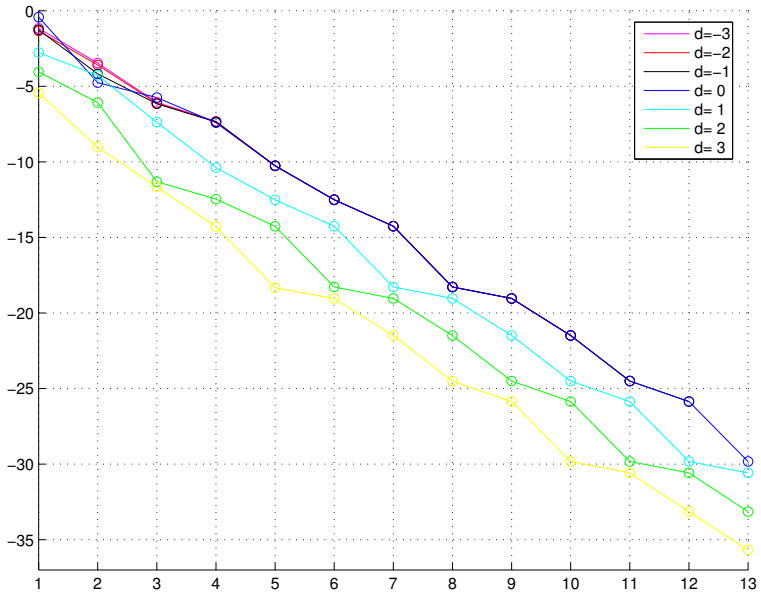


Figure 12: $\ln(1 + r(n, n))$ and friends

Of course L could also be approximated according to its definition as $L(n, n)^{n^{-2}}$ but the above formula offers much better convergence. Using the almost-square legal counts in Table 5.11, as computed by our algorithm, our best estimates using $L(19, 19)$, $L(19, 18)$, and $L(18, 18)$ are

$$L \approx 2.975734192043357249381,$$
$$B \approx 0.96553505933837387,$$
$$A \approx 0.8506399258457145.$$

Table 6 shows the rapid convergence of $L(n, n)L(n + 1, n + 1)/L(n, n + 1)^2$.

Although the formula for $L(m, n)$ is only asymptotic, the convergence turns out to be quite fast. Compared to the exact results in Table 5.7, it achieves relative accuracy 0.99993 at $n = 5$, 0.99999999 at $n = 9$, and 1.00000000000023 at $n = 13$. It is consistent with all the simulated results. For $n = 99$ it gives the same result of $4 \cdot 10^{4638}$. Accuracy is also excellent far away from the diagonal. E.g., at $L(7, 268)$, the relative accuracy is still 1.0000007, witnessing the wide range of application of Theorem 6.

6 Comparison with other games

In Go, the question of what constitutes a legal position is very easy to decide. Not so in some other popular games. In Chess there are so called problems

n	$L(n,n)L(n+1,n+1)/L(n,n+1)^2$
1	2.28
2	3.0
3	2.979
4	2.9756
5	2.975732
6	2.9757343
7	2.9757341927
8	2.9757341918
9	2.975734192044
10	2.975734192044
11	2.975734192043350
12	2.975734192043355
13	2.97573419204335727
14	2.975734192043357255
15	2.97573419204335724932
16	2.975734192043357249362
17	2.9757341920433572493811
18	2.97573419204335724938097

Table 6: Convergence to the base of liberties L.

of *retrograde analysis*, which ask how a position could have possibly arisen. Anyone having tried these problems can appreciate their complexity. But even in as simple a game as Connect-4, the problem of deciding if a position could have legally arisen, can be shown to be NP-hard [12]. In such cases we cannot hope to find exact counts and need to settle for upper bounds.

For Chess, a straightforward Huffman encoding (ignoring history attributes such as castling rights) gives an upperbound of $2^{32\cdot 1+2(8\cdot 3+6\cdot 5+2\cdot 6)} = 2^{164} \sim 2.3\cdot 10^{49}$ positions, while a more refined counting method [4] gives an upperbound of $4.5 \cdot 10^{46}$, less than half of $L(10,10)$.

For Connect-4 played on a board with n columns of height m, each column can be encoded in $m+1$ bits, giving an upperbound of $2^{n(m+1)}$ positions. For the standard board size of 7x6, Edelkamp and Kissmann obtained the exact count of 4531985219092 [15], about 11 times $L(5,5)$.

Gomoku, also known as Go-bang, is more closely related to Go. The goal in Gomoku is to obtain 5 consecutive stones of one's color along an orthogonal or diagonal line. We could thus define a legal position as one in which no 5 same colored stones appear in a row (ignoring the required near-equality of numbers of black and white stones, or allowing the option of passing). This could be computed exactly with a Dynamic Programming algorithm, with states comprising a 4 row thick border. The resulting state space size of 3^{4m} however limits the approach to $m = 6$ or so. The alternative of using the inclusion-exclusion formula to count the illegal positions is similarly limited, since a $6 \times$

$m \setminus n$	1	2	3	4	5	6
1	1	9	907	2098407841	$\sim 10^{31}$	$\sim 10^{100}$
2		386356909593	$\sim 10^{86}$	$10^{\sim 5.3 \cdot 10^2}$		
3			$10^{\sim 1.1 \cdot 10^3}$			

Table 7: Exact and estimated number of games on small boards.

6 board already has 32 possible 5-in-a-rows, all subsets of which have to be considered.

We note that in contrast to Theorem 4, the function $\ln L(m, n)$ for Gomuku is *subadditive*, from which Theorem 5 can be similarly derived. Under a naive assumption of independence of events, the "base of gomoku" would have a value of $1 + 2(1 - \frac{1}{3}^4)^4 \sim 2.90$, much less than for Go.

7 Counting games

7.1 Exact values

By Lemma 2, the number of games equals the number of simple paths in the game graph. For very small boards, we can find these numbers by brute force enumeration, as shown in Table 7.

7.2 Approximation

Just as with legal positions, we can estimate the size of the game tree, and hence the number of games, using Heuristic Sampling. A natural stratifier, similar to the one used by Pang Chen for uncrossed knight tours, is the number of positions reachable through *currently unvisited* positions. Using this stratifier, we obtained the estimates shown in Table 7, with the location of the $\sim$-symbol indicating (un)certainty about the order of magnitude.

7.3 Upper bounds

We can relate the number of simple paths to the product of outdegrees. First we need a technical lemma.

LEMMA 16 *On boards larger than* 1×1*, every node in the game graph has outdegree at least 2.*

PROOF. For the empty position, the Lemma holds by assumption on board size. For other nodes, consider one of its strings. If this string has at least two liberties, then these provide two legal moves for its owner. If it has only one liberty, then the opponent can move there to capture, while a move by the owner there cannot result in a single-stone suicide. •

$m \setminus n$	1	2	3	4
1	0	2.4	2.8	3.512
2		3.368	4.728	6.208
3			6.801	8.933
4				11.741

Table 8: Average outdegree on small boards.

Now consider the game tree, consisting of all simple paths. We want to avoid internal nodes with only one child, so we make them binary by duplicating their child subtree. By Lemma 16, the resulting tree still has no more than $(\prod_v \text{outdeg}(v))/\text{mindeg}$ leaves, where mindeg ≥ 2 is the minimum outdegree. Furthermore, since the tree is at least binary, it must have fewer internal nodes than leaves. This proves

LEMMA 17 *The number of games on an $m \times n$ board with, $mn > 1$, is at most $\prod_v$ outdeg(v).*

By Corollary 1, this is in turn bounded by $(2mn)^{L(m,n)}$. Most positions have about $mn/3$ empty points though, and some of the moves are illegal self-loops, so the average outdegree is much less than $2mn$.

THEOREM 7 *The number of games on an $m \times n$ board is at most $(mn)^{L(m,n)}$.*

PROOF. The Theorem holds trivially for $mn = 1$, which has only 1 position and 1 game, and for $mn = 2$, which has only 5 positions and 9 games. Table 8 shows that for other small boards, the average outdegree is smaller than mn. By the Chernoff bounds on binomial tails, the fraction of legal positions with at least $mn/2$ empty points diminishes exponentially on larger boards, for which the average outdegree is close to $2mn/3$. To complete the proof, note that the product of outdegrees equals the $L(m,n)$'th power of their geometric average, which is upperbounded by the arithmetic average. •

This bound is quite crude for small boards. For example, the 1×3 board has an average outdegree of $\frac{42}{15} = 2.8$, an outdegree product of $3 \cdot 2^{19} = 1572864$ which is bounded by $3^{15} = 14348907$, while the actual number of games is only 907.

We conjecture that for any $mn \geq 3$, the number of games is less than $(2mn/3)^{L(m,n)}$. The fact that legal positions have on average more empty points than arbitrary positions should be amply offset by the removal of self-loops and, more importantly, the widening gap between geometric and arithmetic averages.

7.4 Lower bounds

Note that the game graph need not be Hamiltonian, and constructing a game visiting even 2^{mn} legal positions is a major challenge (achievable for one-dimensional

boards as we'll see later). We can still get a nontrivial lower bound by visiting only a highly structured subset of legal positions.

THEOREM 8 *Suppose the mn points on the board can be partitioned into 3 sets B, W, E such that*

使用格雷码构造棋局

- $|B| = |W| = k, |E| = l = mn - 2k$,
- *B and W are connected,*
- *each point in E is adjacent to both B and W.*

Then there are at least $(k!)^{2^{l-1}}$ possible games, all lasting over $k2^{l-1}$ moves.

PROOF. First we recall some properties of binary *Gray* codes. The 1 bit Gray code G_1 is $0, 1$, while G_{l+1} consists of G_l in which each bitstring is extended with a 0, followed by the reverse of G_l in which each bitstring is extended with a 1. Thus, $G_2 = 00, 10, 11, 01$ and $G_3 = 000, 100, 110, 010, 011, 111, 101, 001$. Successive strings differ in exactly one bit, and the sequence of bits flipped is the so-called *binary carry schedule* $0, 1, 0, 2, 0, 1, 0, 3, 0, 1, 0, 2, \ldots$. Consider the successive changes in Hamming weight. Denote an increment by $+$ and a decrement by $-$. Since the initial bit is flipped every other step, every $4i$-th sign is $+$ while every $4i+2$-nd sign is $-$. As a consequence, G_l has $2^{l-1} - 1$ sign changes. Furthermore, the initial bit is 1 inbetween a change from $+$ to $-$, and is 0 inbetween a change from $-$ to $+$.

Let $\{E_i\}_{0 \le i < 2^l}$ be the sequence of all subsets of E corresponding to the l-bit Gray code. Let E_i^+ be the position with only black stones on $B \cup E_i$ and E_i^- the position with only white stones on $W \cup (E \setminus E_i)$. By the connectivity assumptions, these positions have only one string. See Figure 13 for an illustration of the 3 bit Gray code and corresponding positions.

We now construct games as follows: first play from the empty board to E_0^+ in one of $k!$ ways. Then a black move takes us to E_1^+, and another takes us to E_2^+. Since the sign changes to $-$, we first move from E_2^+ to E_2^- by letting white occupy all of W in one of $k!$ ways and then occupy $E \setminus E_2$, capturing black. Now a white move takes us to E_3^-. In this way we transform each $+$ into a black move, each $-$ into a white move, and each sign change into a capture sequence. We get $k!$ choices at the start, and at every sign change, for a total of 2^{l-1} times.

It remains to show that no position is repeated. All E_i^+, E_i^- positions are unique by construction. During a change from E_i^+ to E_i^-, by the above observation on sign changes, the point corresponding to the initial bit of the Gray code is black, so all intermediate positions can be uniquely associated with E_i^+. Similarly, during a change from E_j^- to E_j^+, that same point is white and all intermediate positions can be uniquely associated with E_i^-. •

COROLLARY 7 *There are between $2^{2^{n^2/2 - O(n)}}$ and $2^{2^{n^2 \log 3 + \log\log n + O(1)}}$ Go games on an $n \times n$ board,*

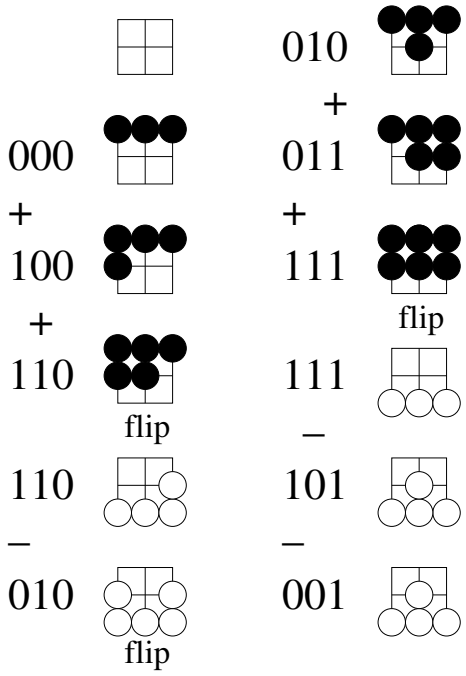


Figure 13: 3-bit Gray code and corresponding positions.

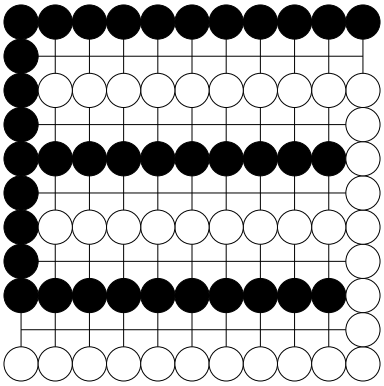


Figure 14: Board partition for games lower bound.

PROOF. On a square board of size $n \equiv 3 \bmod 4$, the partition shown in Figure 14 has parameters $k = |B| = |W| = n - 1 + (n-2)(n+1)/4$ and $l = 2 + (n-2)(n-1)/2$, giving a lower bound of $(k!)^{2^{l-1}} = 2^{2^{n^2/2 - 3n/2 + \log k + \log\log k + O(1)}}$. The upper bound follows from Theorem 7. •

COROLLARY 8 *The number N of 19×19 Go games is*

$$(103!)^{2^{154}} \leq N \leq 361^{0.012 \cdot 3^{361}},$$

in binary

$$2^{2^{163}} < N < 2^{2^{569}},$$

and in decimal

$$10^{10^{48}} < N < 10^{10^{171}}.$$

In one dimension, the conditions of Theorem 8 can only be met by taking E a singleton set, giving a useless bound. Fortunately, the highly structured nature of one-dimensional boards allows us to prove much better bounds.

1路围棋的棋局数量

THEOREM 9 *There are at least $2^{2^{n-1}}$ games on an $1 \times n$ board with $n \geq 2$, which last from $3 \cdot 2^{n-1} - 5$ up to $2^{n+1} - 4$ moves.*

PROOF. Number the points $0, 1, \ldots, n-1$ from left to right. To any non-empty legal position we can assign a *predecessor* in which the leftmost stone, say on point i, is replaced by stones of the opposite color on points $0, 1, \ldots, i-1$. If we assign to each position the number $\sum_{i \text{ occupied}} 2^i$, then the predecessor of a position is indeed numbered one less. The predecessor relation defines a spanning tree of the game graph which we call the *left-capturing* gametree, in which each legal position occurs at a depth equal to its number. Figure 15 shows the left-capturing gametree for $n = 3$. This is a binary tree; each node has 0,1,

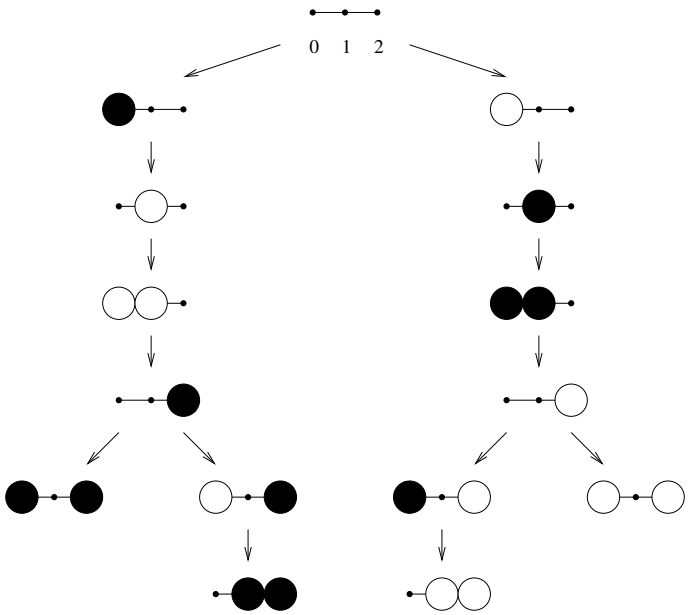


Figure 15: Left capturing gametree.

or 2 children, since at most 2 positions can have the same predecessor. The two positions at maximum depth $2^n - 2$ have a single string occupying all but the leftmost point. The path to such a position has 2^{n-2} positions with point 0 occupied and point 1 empty. Each of these latter positions could be skipped by moving directly from its predecessor to its successor, yielding 2^{n-2} binary choices. To double that amount, we consider games following both paths. For odd $n \geq 3$, we can move directly from the end of one path, e.g. `.XXXX` to the second node `O...` of the other path. For $n = 2$, the Theorem is easily seen to hold. For even $n \geq 4$, we can move from the 3rd-to-last node of one path, e.g. `..XXXX`, to the skippable `X.XXXX`, to the third node `.O...` of the other path. Either case gives us $2^{n-1} - 1$ binary choices, since we lose one choice at the start of the second path. However, we also have a choice of which path to follow first, giving us the required 2^{n-1} choices. The shortest game is reached by skipping all on an even-sized board, while the longest game is reached by skipping nothing on an odd-sized board. •

8 Hamiltonian games

The previous section showed the existence of games visiting a large fraction of all positions. The question arises if and when it is possible to have games encompassing *all* $L(m, n)$ legal positions. Inspired by graph theory, we call such

games, as well as the board they're played on, *Hamiltonian*.

Hamilton 性质

THEOREM 10 *Only one-dimensional boards can be Hamiltonian.*

PROOF. Consider a 2×2 square on an $m \times n$ board with $m, n \geq 2$. There are a total of 10 positions with all points outside the square being black, and having in the square either two opposite black stones or two adjacent black stones and one white stone. Yet there are only a total of 8 positions where they can move to, namely 4 positions with every point except one in the square being black, plus the 4 positions with only two adjacent white stones in the square. •

The question remains which one-dimensional boards are hamiltonian. This means that the graph $G(1, n)$ must have a directed Hamiltonian path starting at the empty position. While $G(1, 1)$ trivially satisfies this, the graph $G(1, 2)$ in Figure 1 does not, since removing the starting node disconnects it.

One sufficient condition for such a path is for the graph minus the empty node to have a directed Hamiltonian cycle, since we can break the cycle at any 1-stone position and insert the empty node there. An even stronger condition is to have each position opposite its reversed color version on the cycle. It turns out that all cycles on $G(1, 3)$ and $G(1, 4)$ are of this form; it suffices to show only half of the cycle, from which the other half can be obtained by reversing colors:

```
O..  O.O  .X.  XX.  ..O  X.O  .OO

O...  O..O  O.X.  .XX.  XXX.  ...O  X..O  X.X.  X.XX  .O..
.O.X  OO.X  ..XX  O.XX  OO..  OO.O  ..X.  .OX.  .O.O  .OOO
```

For larger graphs it becomes impossible to find such cycles manually, and one has to turn to tools such as Concorde [5], for solving Traveling Salesman Problems.

Here's how we translate our directed color symmetric cycle problem into an undirected TSP. For each non-empty node $v \in G(m, n)$ whose leftmost stone is white, we create the the 4 nodes and 4 edges shown in Figure 16. Then, for every move (u, v) where u's leftmost stone is white, we create edges $\{u_0^{\text{out}}, v_0^{\text{in}}\}$ and $\{u_1^{\text{out}}, v_1^{\text{in}}\}$ if v's leftmost stone is white, or edges $\{u_0^{\text{out}}, w_1^{\text{in}}\}$ and $\{u_1^{\text{out}}, w_0^{\text{in}}\}$ if v's leftmost stone is black and w is its color reversed version. All these intra-gadget edges have cost 2. Finally, we cross the pair of 1-edges in one gadget. Call the resulting weighted undirected graph on $4(L(m, n)-1)/2$ nodes $T(m, n)$.

LEMMA 18 *$G(m, n)$ minus the empty node has a color symmetric cycle if and only if $T(m, n)$ has a cycle of cost $3(L(m, n) - 1)/2$.*

PROOF. Each half of a color symmetric cycle in $G(m, n)$ corresponds to a cycle in $T(m, n)$ in which each inter-gadget 2-edge is followed by the top 0-edge, a vertical 1− edge, and finally the bottom 0 edge. The crossed pair of 1-edges ensures proper completion of the cycle. The total cost of this cycle is $2 + 0 + 1 + 0 = 3$ per gadget as claimed. If we attribute to each gadget half the

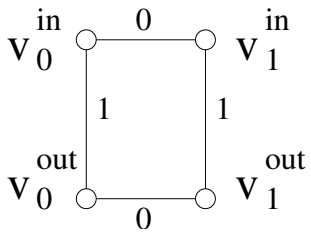


Figure 16: A gadget.

cost of all adjacent traversed 2-edges plus the cost of its traversed 1-edges, then we see that only the above traversal achieves a cost of 3. Thus a cycle of cost $3(L(m,n)-1)/2$ traverses each gadget exactly once, in an order which yields a color symmetric cycle in $G(m,n)$. •

Using this lemma, we found the following 1x5 cycle:

```
O....  O...O  .X..O  .XO.O  O.O.O  O.OOO  .X...  .X.O.  XX.O.
..OO.  O.OO.  .XOO.  .X..X  XX..X  ..O.X  O.O.X  .XO.X  .X.XX
XX.XX  ..O..  .XO..  O.O..  O.OX.  .X.X.  XX.X.  ..OX.  X.OX.
.OOX.  X..X.  X..XX  X.O..  X.O.X  .OO.X  OOO.X  ...XX  .O.XX
.OO..  OOO..  OOOO.  ....X  O...X  O..XX  OO.XX  ..XXX  O.XXX
OO...  OO..X  ..X.X  O.X.X  .XX.X  XXX.X  ...O.  X..O.  X.OO.
.OOO.  .OOOO
```

as well as 1x6 and 1x7 symmetric cycles which are too long to show here.

9 Open problems

1.对于局面数量的渐进，还基于一个猜想
2.1路围棋的 Hamilton 性质
3.棋局数量的上下界（这个问题在另一篇中有改进）

Theorem 6 and its corollaries are contingent on Conjecture 1. Proving this would be important but might require a deep understanding of the structure of the border state graphs and their spectral properties.

Game graphs are an interesting object of study for graph theorists. We conjecture that all $G(1,n)$ with $n > 2$ have color symmetric cycles.

Finally, a significant gap remains in the double exponent between the upper and lower bound on the number of games.

10 Acknowledgements

We are indebted to Martin Müller for suggesting publication of these results, to Piet Hut for extensive commentary on preliminary versions and supporting both the 18x18 and 19x19 computations, to Michael Di Domenico for supporting and helping script the 19x19 computation, and especially to Michal Koucký for the elegant idea of using Chinese Remaindering and extensive help with developing and running the file-based implementations.

References

[1] R. L. Graham, D. E. Knuth and O. Patashnik, Concrete Mathematics, Addison-Wesley, 2nd Ed., 1994.

[2] http://en.wikipedia.org/wiki/Go_(board_game)

[3] http://tromp.github.io/go.html

[4] http://tromp.github.io/chess/chess.html

[5] http://www.tsp.gatech.edu/concorde.html

[6] http://www-neos.mcs.anl.gov/neos/solvers/co:concorde/TSP.html

[7] D. Wolfe, Go endgames are PSPACE-hard, in More Games of No Chance, MSRI Publications Volume 42, ed. R. J. Nowakowski, 2002, 125–136.

[8] D. Lichtenstein and M. Sipser, GO is Polynomial-Space Hard, Journal of the ACM, Vol. **27**, No. 2, (April 1980) 393–401.

[9] P. Chen, Heuristic Sampling: A Method for Predicting the Performance of Tree Searching Programs, SIAM J. Comput. 21(2), 1992, 295–315.

[10] J. M. Robson, The Complexity of Go, Proc. IFIP (International Federation of Information Processing), 1983, 413–417.

[11] R. E. Blahut, Theory and practice of error control codes, Addison-Wesley, 1983.

[12] Gerhard Woeginger, personal communication.

[13] J. Dean and S. Ghemawat, MapReduce: Simplified Data Processing on Large Clusters, Proc. OSDI'04 (Sixth Symposium on Operating System Design and Implementation), 2004, 137–150.

[14] M. Crâşmaru and J. Tromp, Ladders are PSPACE-complete, Proc. 2nd Int. Conf. Computers and Games, Springer-Verlag, 2000, 241–249.

[15] S. Edelkamp and P. Kissmann, Symbolic Classification of General Two-Player Games, Proc. 31st annual German conference on Advances in Artificial Intelligence, 2008, 185–192.