The number of go games is greater than a googolplex

Matthieu Walraet January 3rd, 2016

Introduction

The number of possible go games is very high. For “normal” games, it is often estimated to 10^{800} taking a maximum game length of 400 moves and an average number of 100 legal moves in each position. Of course what is a normal game depends on the skill of players, there is much more possible go games between true beginners playing randomly than between highly skilled players trying to win.

Here we consider every possible go games allowed by the rules, even if it seems very artificial, like a player filling a good part of the board while the opponent is playing only passes, just to be captured and making room for more moves.

A googolplex is the number $10^{10^{100}}$ This term, coined by Edward Kasner in 1920, is known in popular culture as an extremely large number. Such a number is difficult to apprehend and is almost never reached by real world computations. (Of course, there exist vastly greater numbers, for example those you can obtain using Conway chained arrow notation.)

Previous results

The article “Combinatorics of go” [2] by John Tromp and Gunnar Farnebäck is mainly about exact computation of number of go positions, on go boards up to 17x17 dimension. (These results will be used latter.)

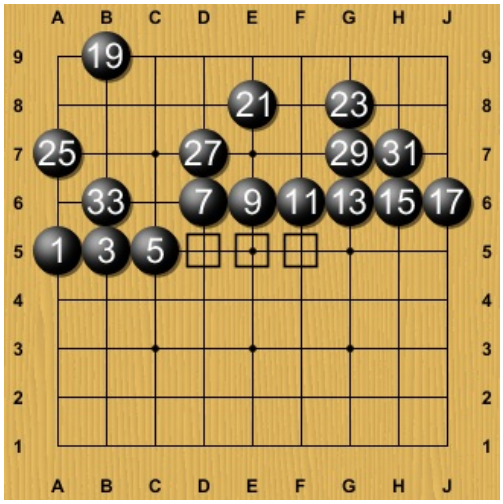
In chapter 6.4, they describe a technique, using properties of binary Grey codes, to visit a high number of legal positions. This gives as lower bounds 10^{48} for game length and $10^{10^{48}}$ for number of possible go games on a 19x19 board.

Main scheme

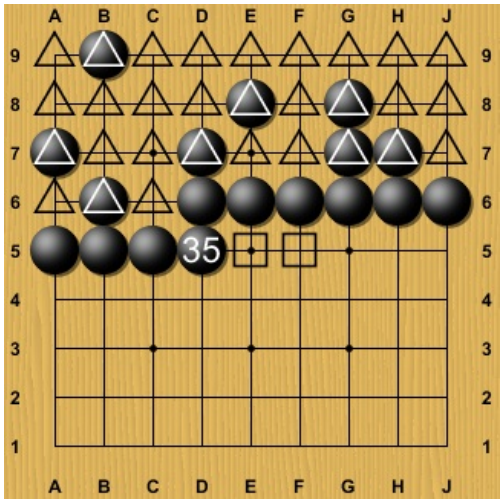
We can compute a lower bound of maximum go game length and of number of possible go game by constructing a go game, and counting all variations allowed by this construction.

When we construct this game, we must ensure it respects the superko rule, that is a position is never repeated. For that, we encode numbers in binary, either with black stones on the upper side or white stone. In the center of the board, 3 intersections signals which side “controls” the board.

+++	Setup phase
●++	Black controls step b
●●+	White controls step a
++○	White controls step b
+○○	Black controls step a

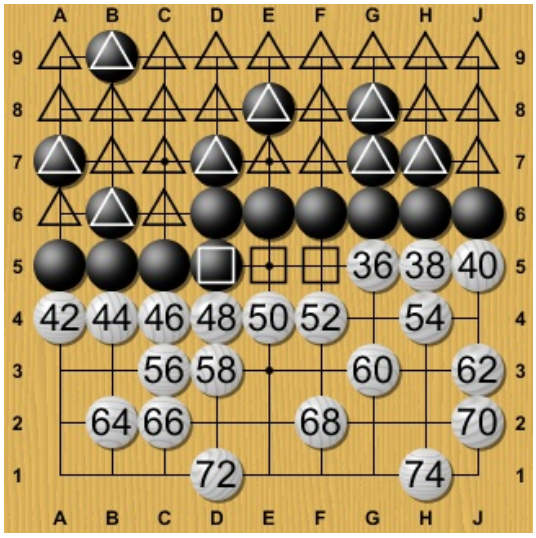


While there is no stone in the three center intersections, no side has “control” yet. As this occurs only at the beginning of the game, there is no repetition possible.



Now, center area contains one stone, black side has control. B_1 is the number encoded in binary by black stones marked with a triangle.

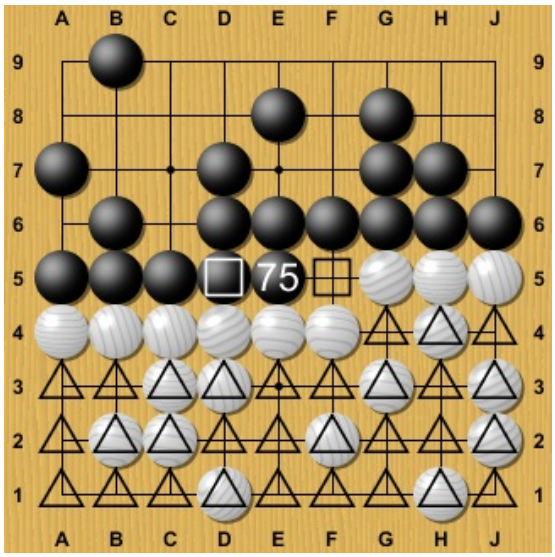
We say the game is now controlled by “B_1 step b”. That means we can play anywhere in the unmarked place (here the bottom side of the board) without risk of repeating a position during the whole game before current control.



Of course we still have to avoid repeating a position during the current control. For now we will just add white stone to encode W_1 number, so there is no stone taken.

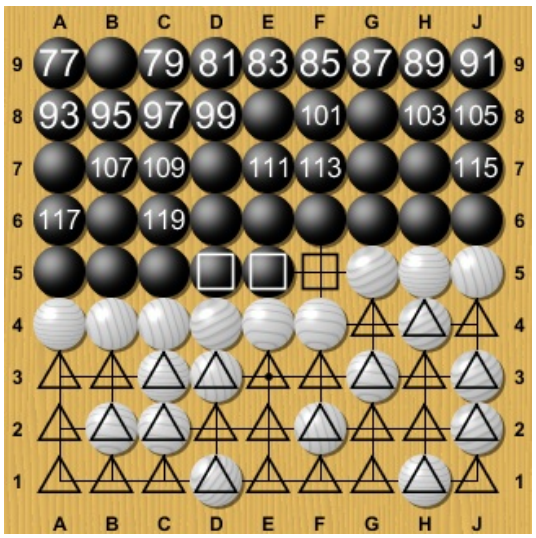
Here is last position under "B_1 step b" control.

This pattern of stones in marked positions (square and triangle) will never occur again in the whole game, ensuring respect of superko rule.

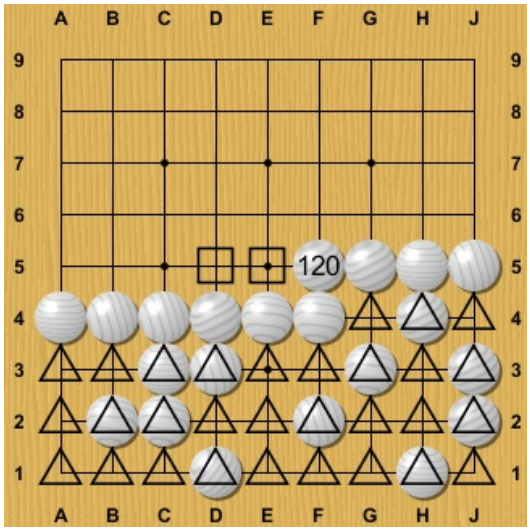


With move 75, we enter in "W_1 step a" control.

We use this control to fill upper side of the board with black stone. We don't care that we may encode numbers with black stones as the center square stones gives control to "white step a"

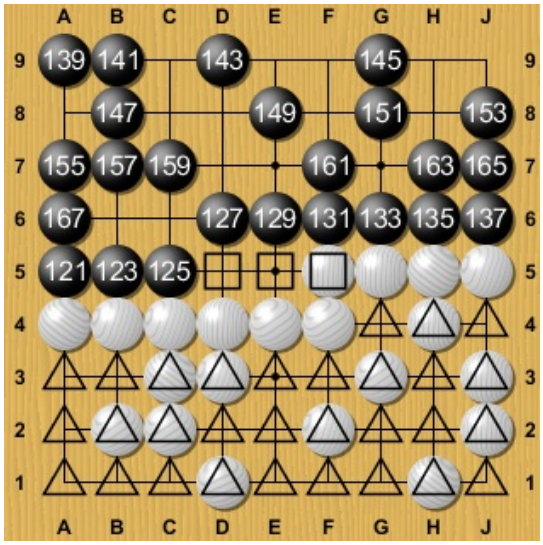


Last position under "W_1 step a" control.

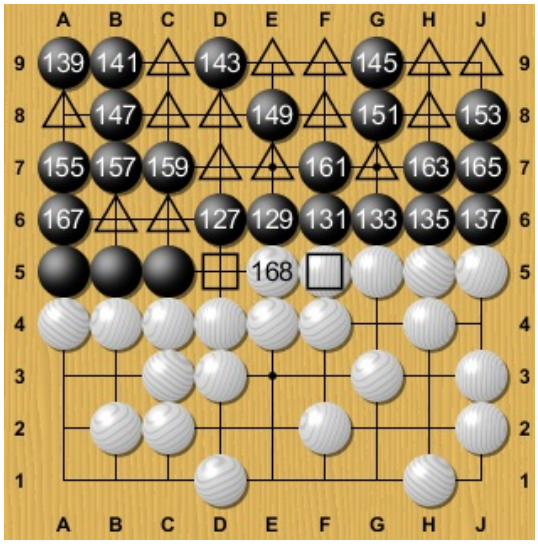


Playing move 120 both takes the stone and changes control to "W_1 step b".

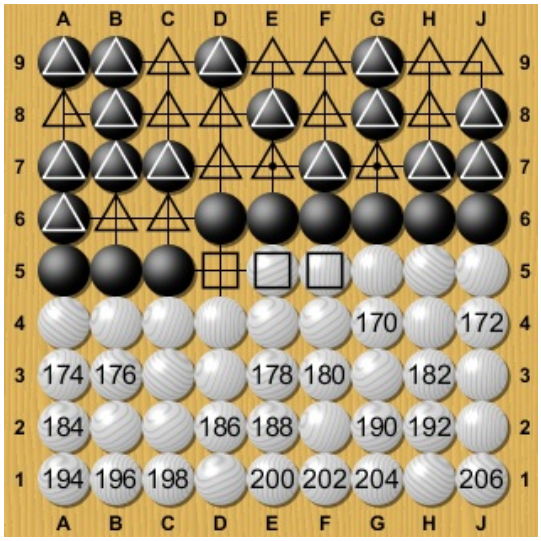
As the configuration of square stones is different, we can play in the upper side and encode B_2 number in black stone. "Step a" and "step b" are two different "controls" even if they use same W_1 number.



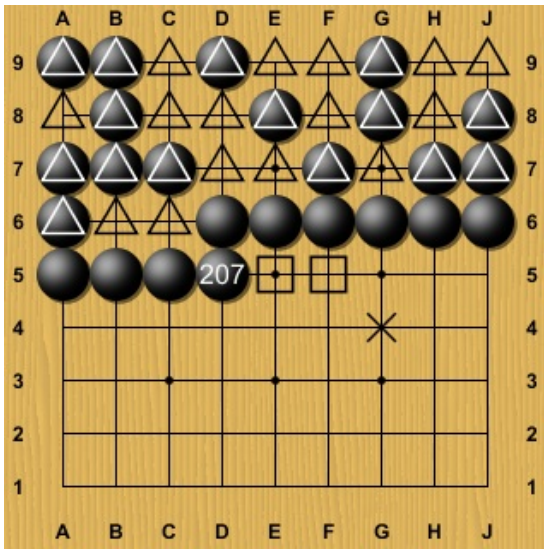
Last move of "W_1 step b".



Playing move 168 starts "B_2 step a" control.



We fill bottom part of the board with white stones.



With move 207, we take them and enters in "B_2 step b" control, starting a new cycle.

We can go on until we run out of numbers. Here on a 9x9 board we have 2^{30} different numbers for each side.

At the end we can even create "B_1 step a" control and fill the white part.

In a board of size NxN, $N \geq 5$, N odd, the number of triangle positions is $\frac{(N+1)(N-3)}{2}$ the usual board 19x19 has 160 of them.

During each cycle, we play N^2+1 stones and N^2-3 passes.

So length of the game is $(2N^2-2)2^{\frac{(N+1)(N-3)}{2}}$

For 19x19 length is 720×2^{160} about 1.05×10^{51}

Number of variations by choosing B_n and W_n number order is: $\left(2^{\frac{(N+1)(N-3)}{2}}!\right)^2$

For 19x19 it is $\left(2^{160}!\right)^2$ about $10^{10^{50.1446}}$ and that's not counting order of playing stones.

We can also vary the this order, in each cycle each colors fills nearly half the board, in two steps each. Only stones played in square marked intersections can't be switched (or only with border stones, but we will ignore that possibility).

We can permute the order of playing the number encoding stones and the border stones during “step b”, and also permute the stones filling the half board in “step a”.

There are N border stones. If half of the digits of the encoded number are 1, the number of way to permute stones playing is $(N+\frac{(N+1)(N-3)}{4})!\times\frac{(N+1)(N-3)}{4}!$

For each encoding number using less than half the stones, there is its opposite, exchanging 0 and 1. We can combine calculation of permutations with these two numbers, and use following inequality: $(N+k-i)!(k+i)!(N+k+i)!(k-i)\geq((N+k)!k!)^2$ with k being the half of the number of digits of the encoded number and i the difference between number of 1 and k.

So the contributions of permutations of stones played during each step, for the whole game is more than:

$$\left(\left(N+\frac{(N+1)(N-3)}{4}\right)!\times\frac{(N+1)(N-3)}{4}!\right)^{2\times 2^{\frac{(N+1)(N-3)}{2}}}$$

For a 19x19 board, its $(99!\times 80!)^{2^{161}}$ about $10^{10^{50.9049}}$

Multiplying also with contributions of B_n and W_n permutations it gives as lower bound for number of possible go games on a NxN board, N odd :

$$\left(2^{\frac{(N+1)(N-3)}{2}}!\right)^2\left(\left(N+\frac{(N+1)(N-3)}{4}\right)!\times\frac{(N+1)(N-3)}{4}!\right)^{2\times 2^{\frac{(N+1)(N-3)}{2}}}$$

For a 19x19 board it gives $10^{10^{50.9744}}$ number of games with this scheme.

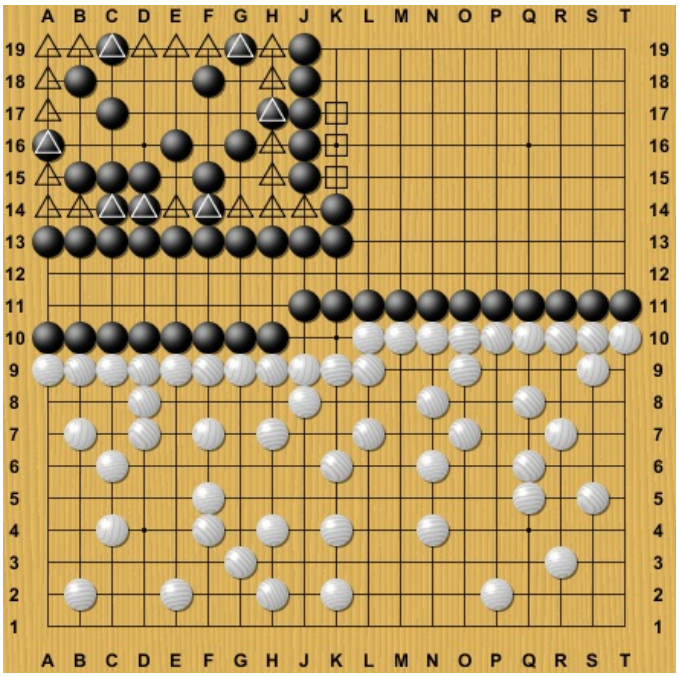
First iteration

Main scheme gives already a good lower bound of the number of go game, but it's only an introduction of how “control” work.

We have seen that when a “control” has started, let's say “W_k step b” has control, we can play anything we want during this control without risking to repeat a previous position of the whole game before this control phase. We must only ensure not to repeat a position within current control.

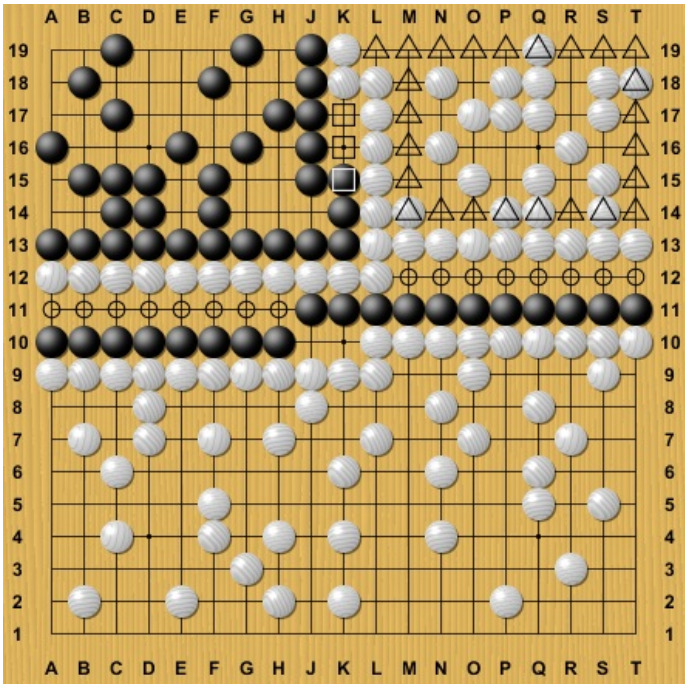
When “W_k step b” begins, the whole upper board is empty. We can repeat the whole “main scheme” on this upper side, encoding numbers in black stones on the upper left and in white stone in upper right and with a “sub-control” area between them.

We have to ensure that we can perform the capture of each quarter of board, and that at the end of the sub-scheme we can resume the main scheme.



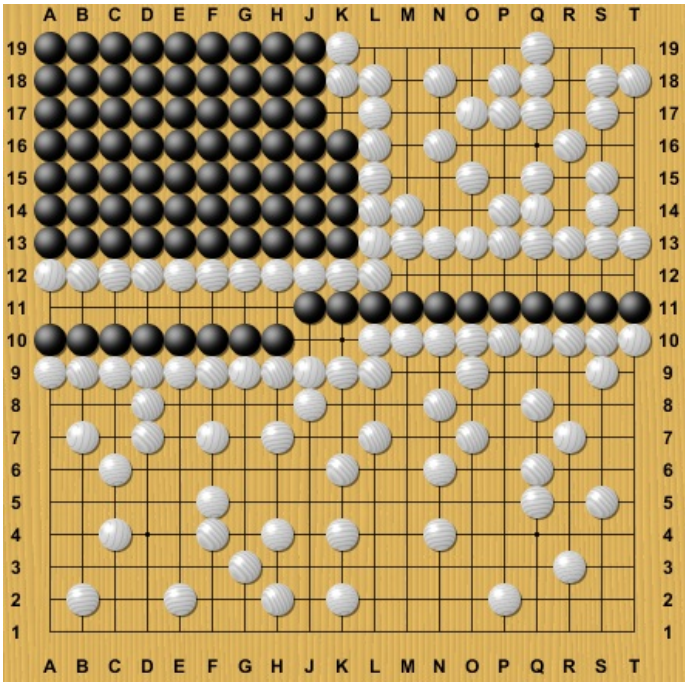
During the setup SW_k subcontrol, we encode number $B_{k,1}$ in triangles area.

There are 49 triangles for 19x19 board.

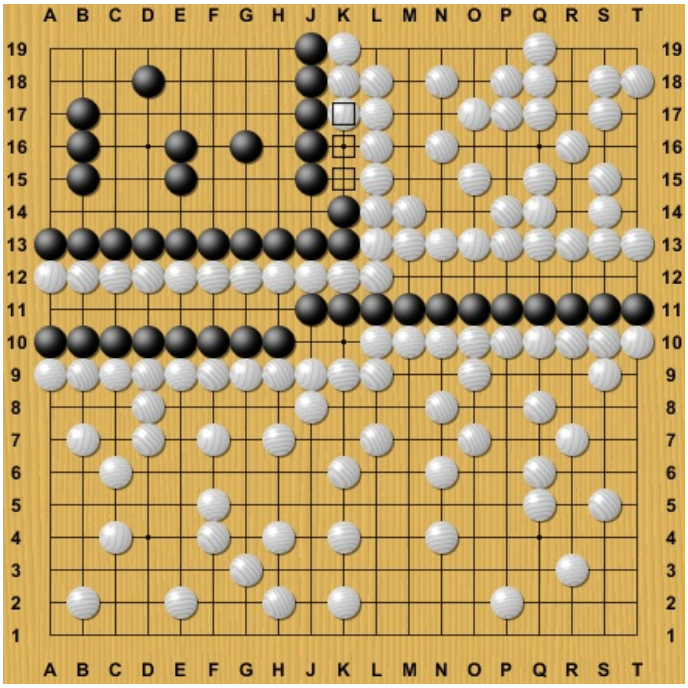


During $B_{k,1}$ step b sub-control, we encode $W_{k,1}$ in white stones.

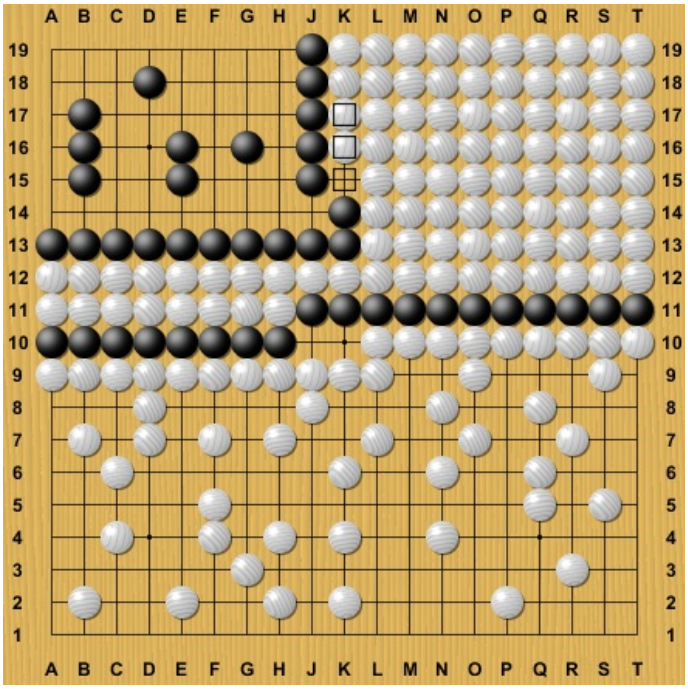
For now we avoid using area marked with circle.



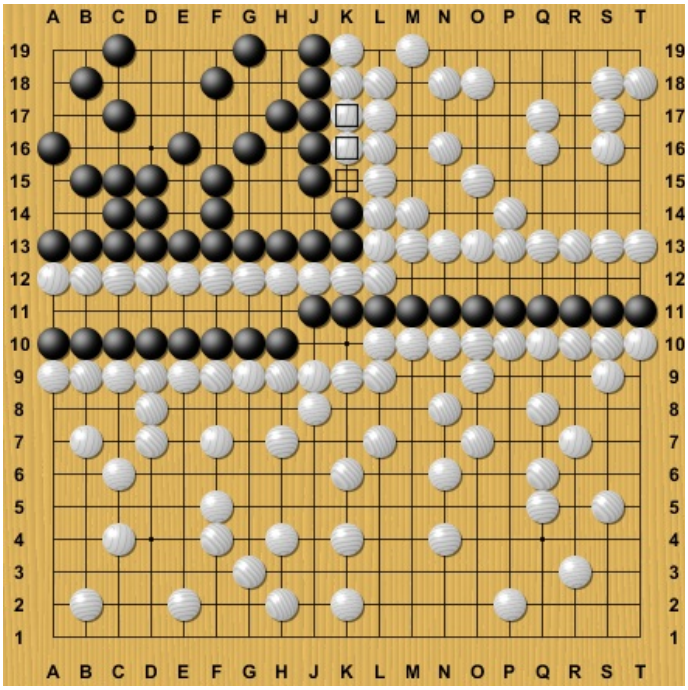
While in $W_{k,1}$ step a sub-control we fill the black side.



While in $W_{k,1}$ step b sub-control, we encode $B_{k,2}$ number.



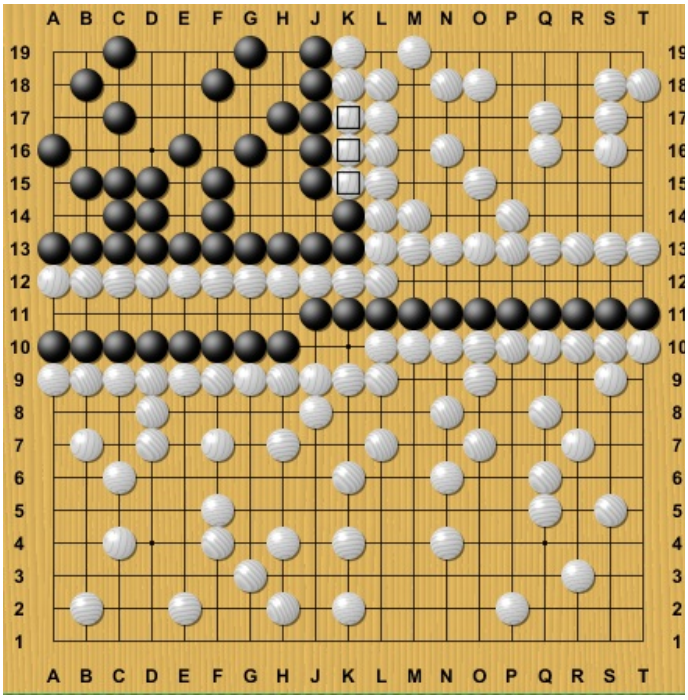
During $B_{k,2}$ step a sub-control we fill the upper right area with stones so they can all be taken and then the cycle goes on.



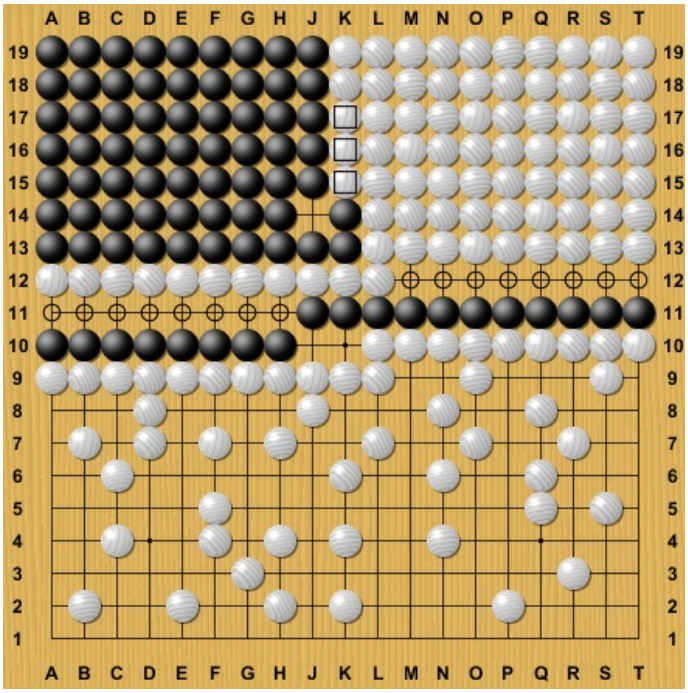
Let's say we are near the end of sub-scheme. We need to be able to re-enter main scheme.

During W k,i step b control, with i=2^49, we have encoded $B_{k,1}$ number again.

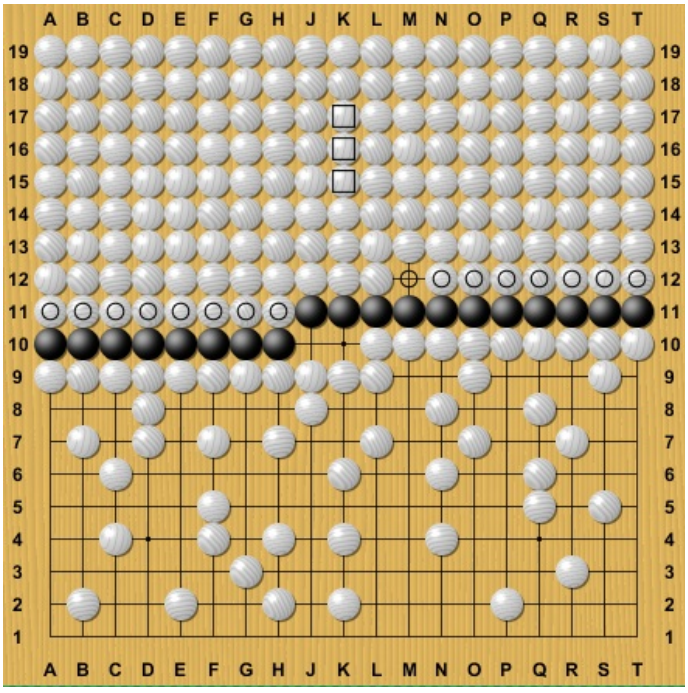
With white K-16, we enter $B_{k,1}$ step a control, we can as very first number starts directly at step b.



Then we directly play K-15
Three stones of the same color in square marked area was and unused code for now, we use it for “clean-up” control phase, the opposite of “setup” control phase.

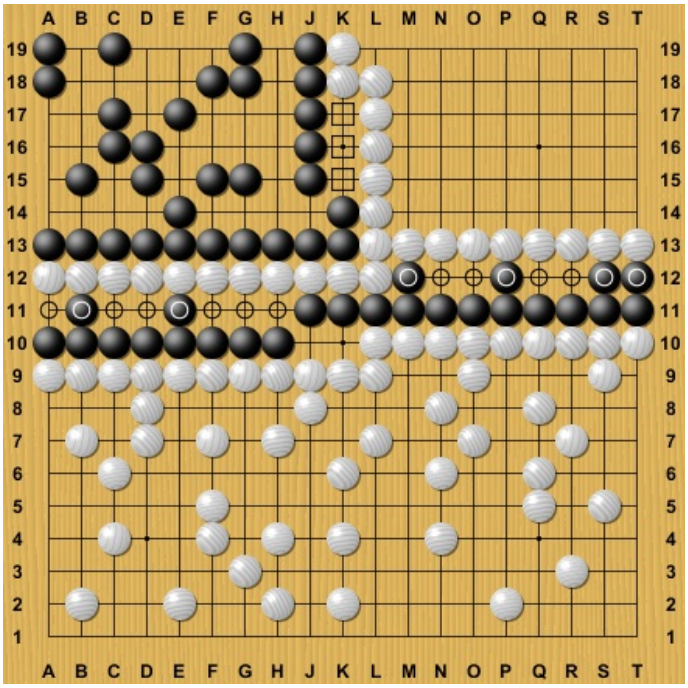


We fill black and white sides, take black, then start filling everything.



Then we fill all the upper side of the board except one of the places marked with circles.

After capture, everything is ready for starting encoding B_{k+1} number in main scheme.
As long as black stone is needed in M12, of course.



However, for each stones in circle area needed to encode $B_{k+1,}$ we can repeat the whole sub-scheme. Every time there is an additional black stone played there, it ensures we will not repeat an already played position.

Here, we are back in the setup phase of the sub scheme. (Playing these white stones in advance is a possible variation.)

Once the last needed black stone in circle area is played, we go back in main scheme.

The sub-scheme can be played also during main scheme setup phase, so we can play it exactly 2^{161} times.

For some numbers of the main scheme, 1 over 2^{16}, the sub-scheme will not be possible at all. In the whole game, as all numbers are used, we can play the sub-scheme a number of times in average exactly half the number of circles (here 16).

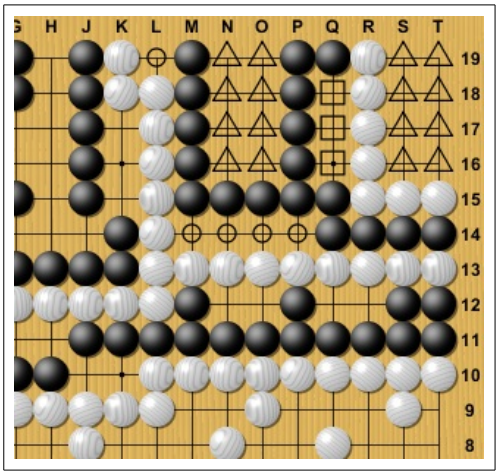
During a white + black step in a sub-scheme, we play full semi-left side, plus semi-right side, plus half of circle, in total 9 board lines, 171 stones.

Number of “black + white” steps in sub-scheme is 2^{161} (number of control phase in main scheme) times 2^{49} (49 triangles area in sub-scheme) times 16/2 (half the number of circle mark).

This gives a new lower bound for the length of the game in 19x19:

171×2^{213} about 2.25×10^{66}

Second iteration



Each corner in sub-scheme has room for a sub-sub-scheme. Even is there is not much place left (8 triangle place) it worth a lot.

There are 2^{214} sub-control.

So black+white sub-sub-control is played $2^{214} \times 2^{8} \times \frac{5}{2}$ times, with 46 stones in average.

Lower bound for the length of the game in 19x19: $46 \times 5 \times 2^{221}$ about 7.55×10^{68}

If we only count permutations of black and white stones during sub-sub-scheme it gives:

(18 black stones + 4) ! 4 remainings! (7 white stones + 4)!, (4 remaining) !, all that power the number of black+white sub-sub-controls.

$\left(11!\,4!\,22!\,4!\right)^{\left(5 \times 2^{221}\right)}$ about $10^{10^{68.72}}$

Ternary scheme

Encoding control number in binary in not the most efficient. An intersection has three possible states, so we can a ternary encoding. As in go each group of stone must have at least one liberty, only a part of these numbers corresponds to a valid positions.

Playing ternary mode

We can play the main scheme, sub-scheme and sub-sub-scheme in ternary mode nearly the same way it was done in binary mode.

During a “step a phase”, playing the needed stones in any order is valid, as final position is valid. No group of stones is taken so there is no risk to repeat the position within current control.

During either a “step b phase” or a “clean-up phase”, it's less obvious as we must play to fill an area containing an arbitrary pattern of black and white stones. We can avoid repeating the position by following this algorithm: (Assuming we have to fill the area with black stones)

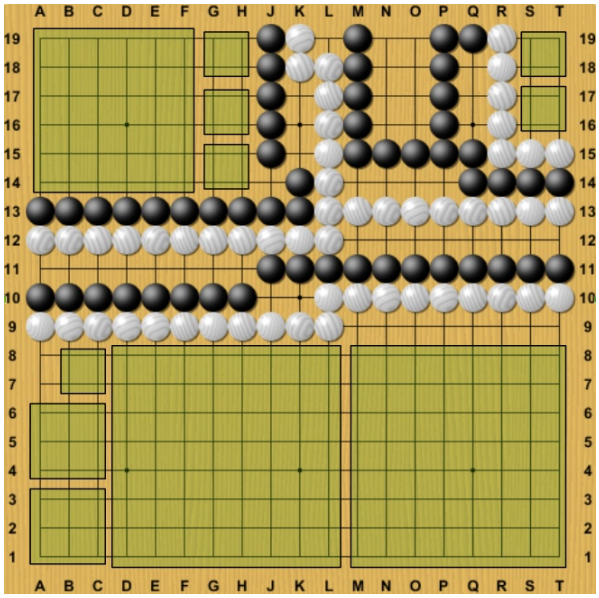
1. Play all valid black moves from the border groups, except the intersection that we keep to take all the stones at the end (marked square or circles). In the case of main scheme step b phase, there can be two border groups sharing the same marked square liberty, in other cases there is only one. White stones can be taken doing this.
2. When there is no more valid black move on border groups, either the area is filled with black (so it's done we can take them all) either there is one or more white groups in the area in contact of border groups. Choose one of these white group.
3. While this white group has more than two liberties, play a white stone in one. Black stones can be taken doing this. When there is only one liberty left, play it with black to capture. Resume to step 1. It will at least fill the area formerly occupied by white group just taken.

While repeating the cycle, the black border group only grows. The capturing black move at the end of step 3 can't break superko rule, as it takes white stones that were in contact of black border group.

We will use this result from [2], that gives the exact number of valid go positions for go board up to 17x17.

Board size n x n	Number of valid go position
1	1
2	57
3	12675
4	24318165
5	414295148741
6	62567386502084877
7	8367784784798428762859 5
8	990966953618170260281935463385

For now we will only do a rough computation, dividing the areas in squares.



Two valid 8x8 positions side by side is a valid position in 16x8.

Main scheme 2 8x8 2 3x3 1 2x2 10 1x1

$990966953618170260281935463385^2$ X

$12675^2 \times 58 \times 2^{10} = 6.888 \times 10^{78}$

Sub scheme 1 6x6 3 2x2 1 1x1

$62567386502084877 \times 58^3 \times 2 = 2.441 \times 10^{22}$

Sub-sub-scheme 2 2x2

$58^2 = 3364$

(We can use 58 instead on 57 for 2x2, as they are always connected to existing stones, so the full 2x2 board in same color is valid)

When numbers where encoded in binary, a circle position could be used half the time. Here, the part of circle position encoded in ternary can be used less than third of the times, but more than a quarter of the time.

So sub-scheme can be repeated 8/2 + 8/4 = 6 times per main scheme control, 12 times per cycle.

Sub-sub-scheme can be repeated 1/2 + 4/4 = 3/2 times per sub-scheme control, 3 times per cycle.

Number of times “black+white sub-sub-control” is played:

$$(6.888 \times 10^{78}) \times 12 \times (2.441 \times 10^{22}) \times 3 \times 3364 = 2.036 \times 10^{106}$$

If we keep (for now) 46 stones played in average, it gives as lower bound for maximum game length on a 19x19 board: 9.366×10^{107}

For number of game, we count only permutations within all “black+white sub-sub-controls”.

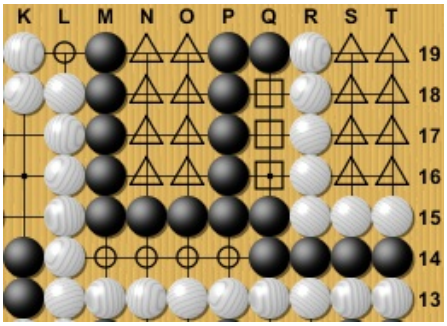
And for number of permutation per cycle if we also keep for now the way it was computed previously (it's a low estimate) : 11! 4! 22! 4!

$$(11!\,4!\,22!\,4!)^{2.036 \times 10^{106}} \quad \text{about} \quad 10^{10^{107.8}}$$

The number of permutations of the “main scheme” and of the “sub-controls” are small relative to this number, so we can ignore their contributions.

Refining computations

The last steps, the sub-sub-control, is very important for the overall figures. We can write a program to compute more precise numbers. [3]



A 2x4 go board has 4125 possible positions.

In the corner, we can use more positions, as white stones on S column and row 16 can't be taken. This gives 4779 positions.

In the left triangles positions, all black stones don't need internal liberties. This give 4996 different positions. However, in the corner scheme we can't use more positions here than in corner.

In corner:

For each possible position, there are different ways to obtain it on the board (“step b” part) and then from this to fill the area in white stones (“step a” part). Some stones can be taken in the process, either for “step a” or “step b”, as long as there is no repetition. To ensure the superko, if border is white (as on figure above) we allow only black stones to be captured.

For “step a” if we already are in the target position (all in white stones) there is only 1 way to obtain it and its length is 0. For the other positions, we can compute number of ways and maximal length recursively by considering all white moves and all black moves that don't take any white stone. Number of ways is the sum of the ones of next moves one. Max length is one plus the maximum of max length of all next moves. This is quite fast thanks to memoization.

For “step b”, we can use the same technique, as long as we are careful that some branch of recursion are dead ends (black stones played that can't be taken without unwanted white ones.) However, this can be quite slow, as memoization has to restart for each considered target position.

Another solution is to go backward. For each given position we compute recursively the number of ways (and max length) to reach the starting (here empty) position by playing backward.

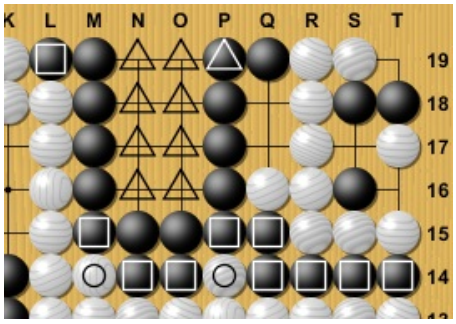
As during the sub-sub-scheme we plays “step b” and “step a” for all 4779 positions, we add all the length and multiply all the number of ways, giving for the corner part.

Maximum length (not counting passes): 235427

Number of games: 1.07×10^{119526}

In the other part:

We could use the same technique as before, with a little optimization.

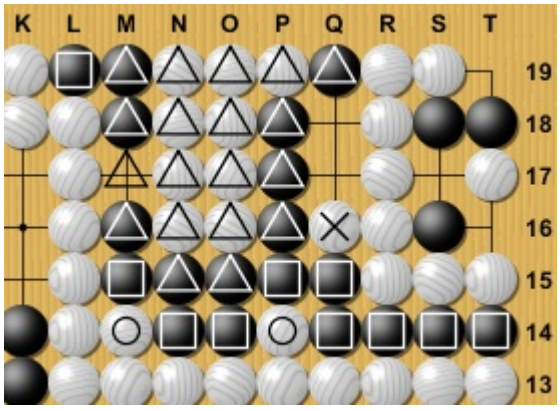


You can play white stones only on triangle places (including P19)

All square marked black stones can be considered together in the recursion. You only consider the number of stones in the pool, and when you play the pool, you count there is K ways to get to the position with K-1 stones in the pool.

However, even with this optimization, computation is outside the reach of my current program.

So to help, we consider above diagram to be the starting point of the program, and compute manually how to get there. Of course some possibilities are missed.



We can play the triangle stones (black and white) in any order, then take the 9 white stones, with black M17 (or choose another of the 10 black stones). Then play 8 white stones in columns N and O, to take them with black P19.

Length is 18+1+8+1 = 28

Number is $18!8!10$

We can place the square marked stones anywhere in this sequence of 28 moves. If there are k squares, first one has 29 possible place, last one has 28+k.

Game length is 28+k Number of games: $18!8!10\frac{(28+k)!}{28!}$

There are more possibilities with less white stones played. For now we won't try to count them.

For this case, with two white stone marked circles played, and so 10 square.

Max length of game: 38 Number of games: $18!8!10\frac{38!}{28!}=4.428\times10^{36}$

Number of circles:

In the worst case, there is 5 white stone at the end of sub-sub-scheme played, but this mean we have played the sub-sub-scheme 5 times, first one with 0 circle white stones (12 square black stones), and last one with 4 circle white stones (8 square black stones). For each case where there is 8 squares, there is at least one case where there is 12. Idem for resp 9 and 11. So we can use the 10 value.

Using the intermediate position, with all black stones placed as starting position, the program can compute the max length and number of games for the 4996 possible positions, and select the 4779 best one (as we can't play more). This gives:

Maximum length: 166155

Number of games: 4.191×10^{63224}

We have also to count 4779 times the “manual computation” of how to go from empty position to the intermediate one.

Maximum length (not counting passes): $4779\times38=181602$

Number of games: $\left(18!8!10\frac{38!}{28!}\right)^{4779}=2.436\times10^{175132}$

This gives for the whole ternary sub-sub-scheme

Maximum length (not counting passes): 583184

Number of games: 1.0923×10^{357883}

From previous chapter, number of times “sub-sub-scheme” is played:

$\left(6.888\times10^{78}\right)\times12\times\left(2.441\times10^{22}\right)\times3 = 6.052\times10^{102}$

This gives for the whole game:

Lower bound for maximum go game length: $583184 \times 6.052 \times 10^{102} = 3.53 \times 10^{108}$

Lower bound for number of possible go games:

$$\left(1.0923 \times 10^{357883}\right)^{6.052 \times 10^{102}} = 10^{10^{108.336}}$$

References

[1] Sensei Library http://senseis.xmp.net/?NumberOfPossibleGoGames (version 43)

[2] John Tromp and Gunnar Farnebäck, Combinatorics of Go, 2009, https://tromp.github.io/go/gostate.pdf

[3] You can find materials used for this article here: http://matthieuw.github.io/go-games-number/

- SGF files used to create go diagrams
- Programs used for computations (in Python)