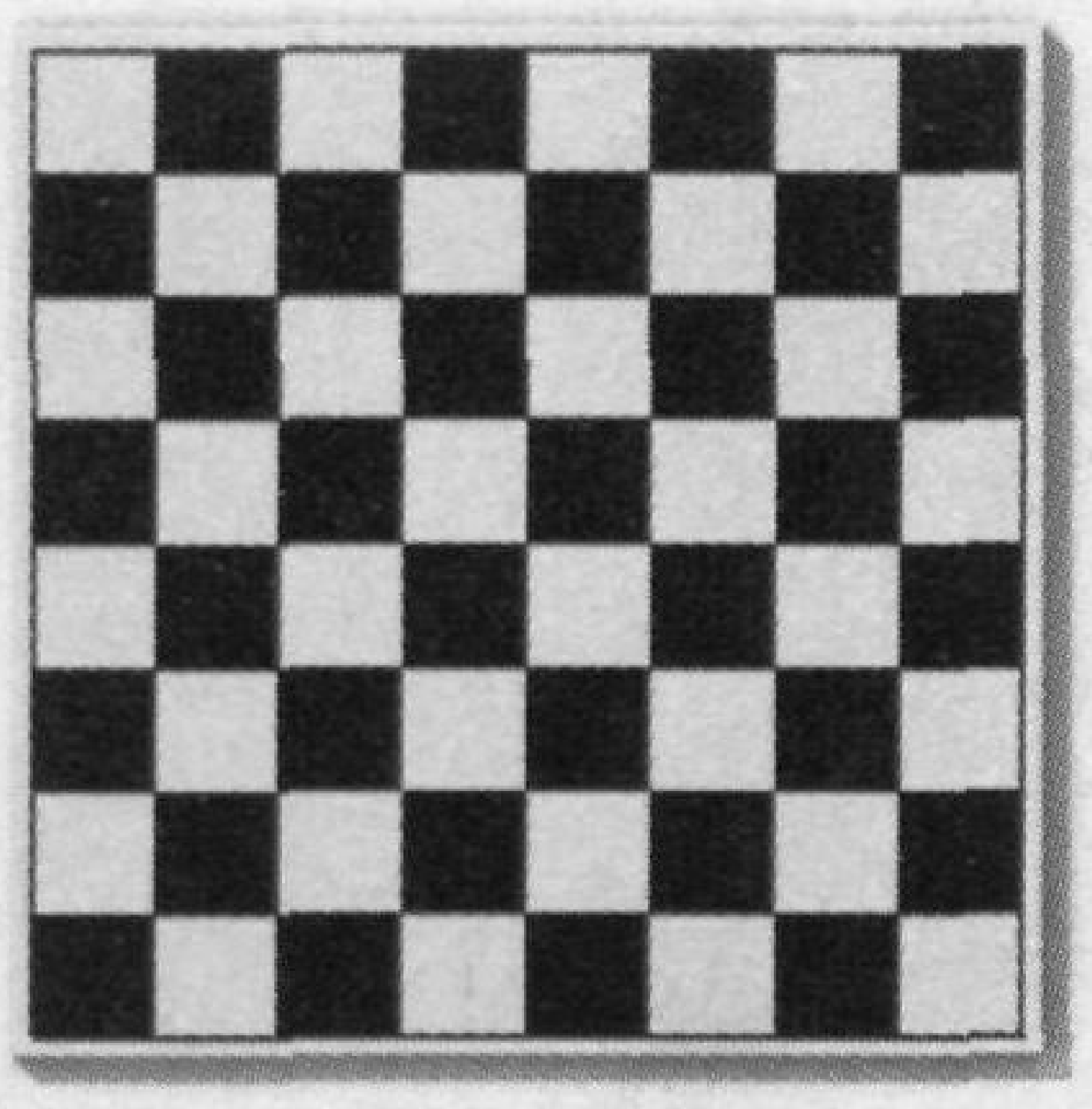
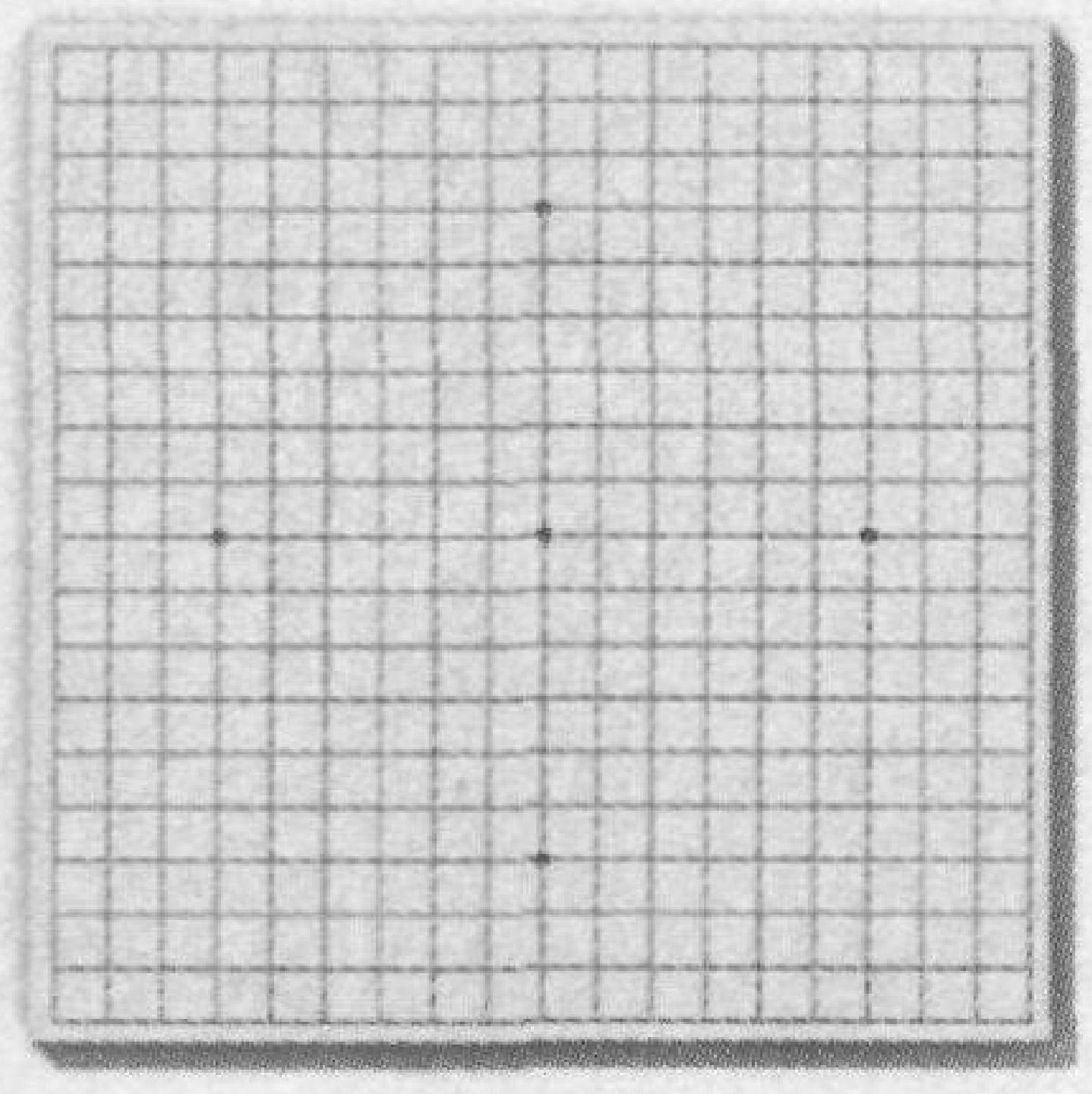
CHESS VS. GO





GRID SIZE

8 x 8	19 x 19

AVERAGE NUMBER OF MOVE CHOICES PER TURN

35	200–300

LENGTH OF TYPICAL GAME

60 moves	200 moves

NUMBER OF POSSIBLE GAME POSITIONS

10^{120}	10^{170}

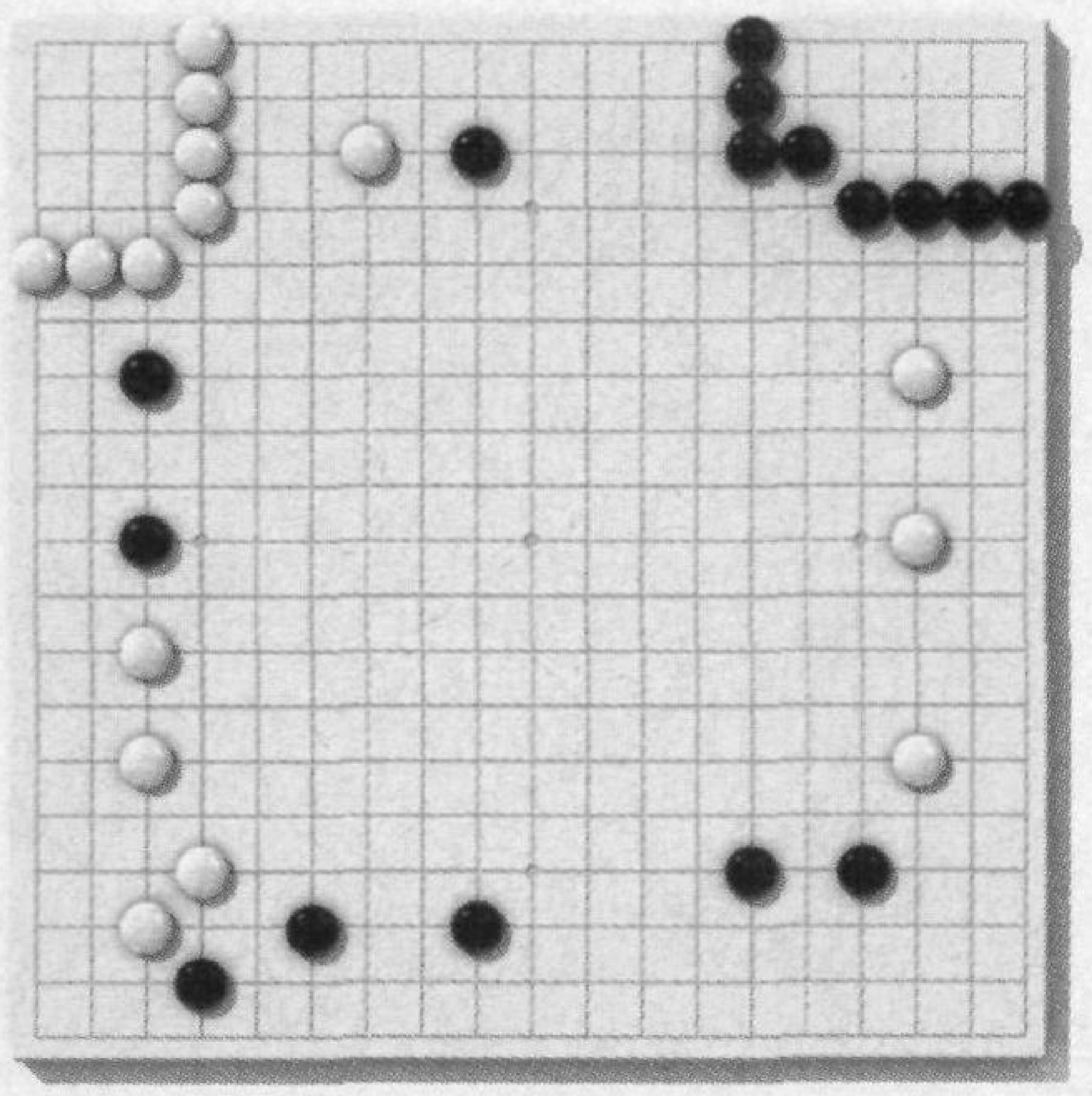
EXPLOSION OF CHOICES
(starting from average game position)

35	Move 1	200
1225	Move 2	40 000
42 875	Move 3	8 000 000
1 500 625	Move 4	1 600 000 000

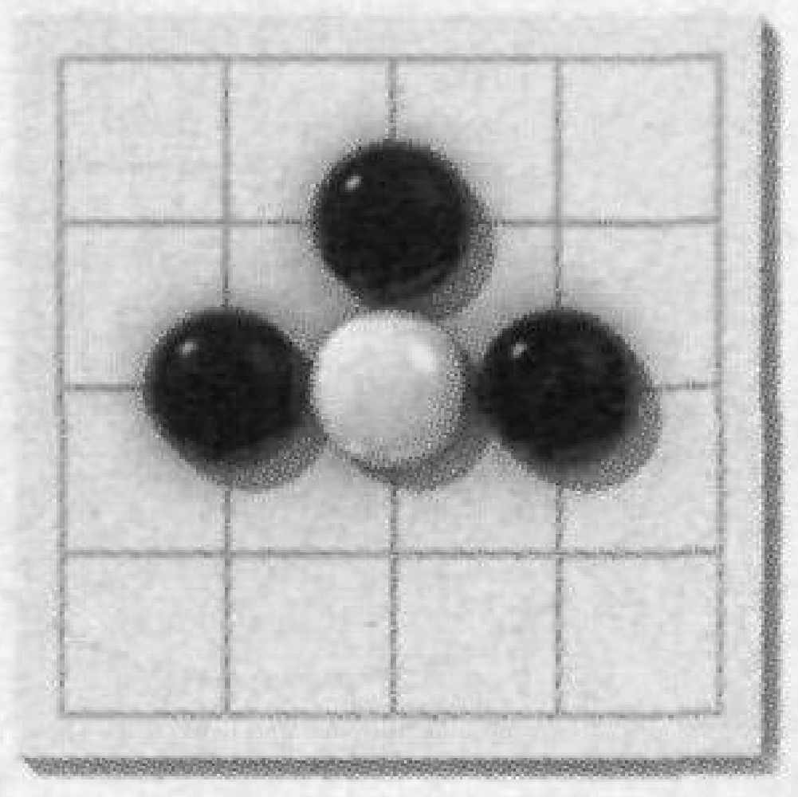
THE GOAL OF GO

The object of Go is to enlarge your territory at your opponent's expense. One way is by surrounding your opponent's stones by putting your own stones on the adjacent points, which are known as "liberties." Once surrounded, stones are removed from the board and become your prisoners, each worth a point.

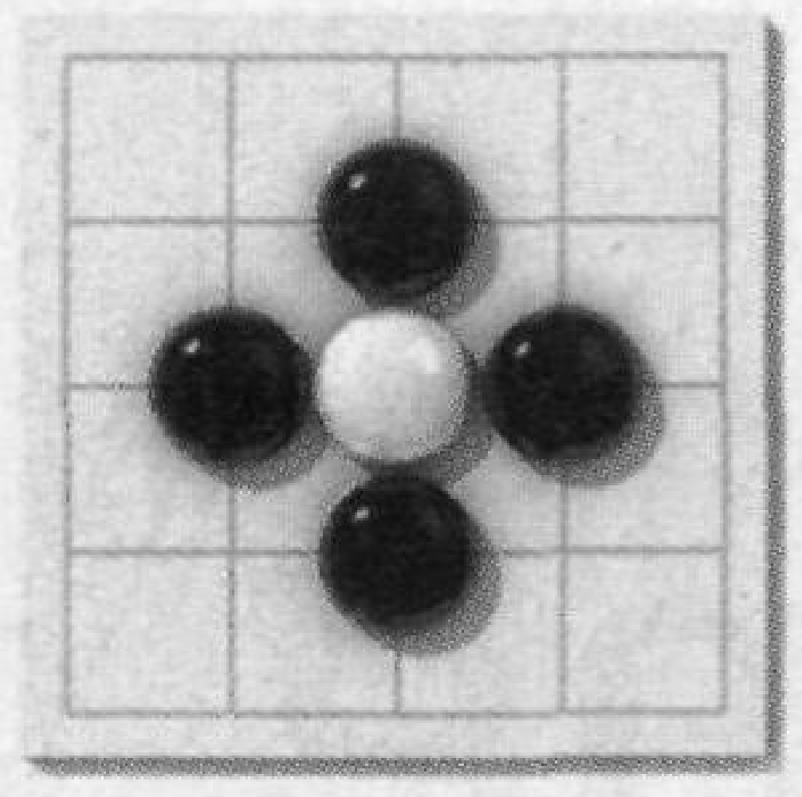
Another way to claim territory is by surrounding empty space—that is, unoccupied intersections, each of which is also worth a point. Here, for instance, the two groups in the corner each enclose nine spaces, worth as many points. Obviously, it takes fewer stones to enclose territory at the corners than in the middle of the board.



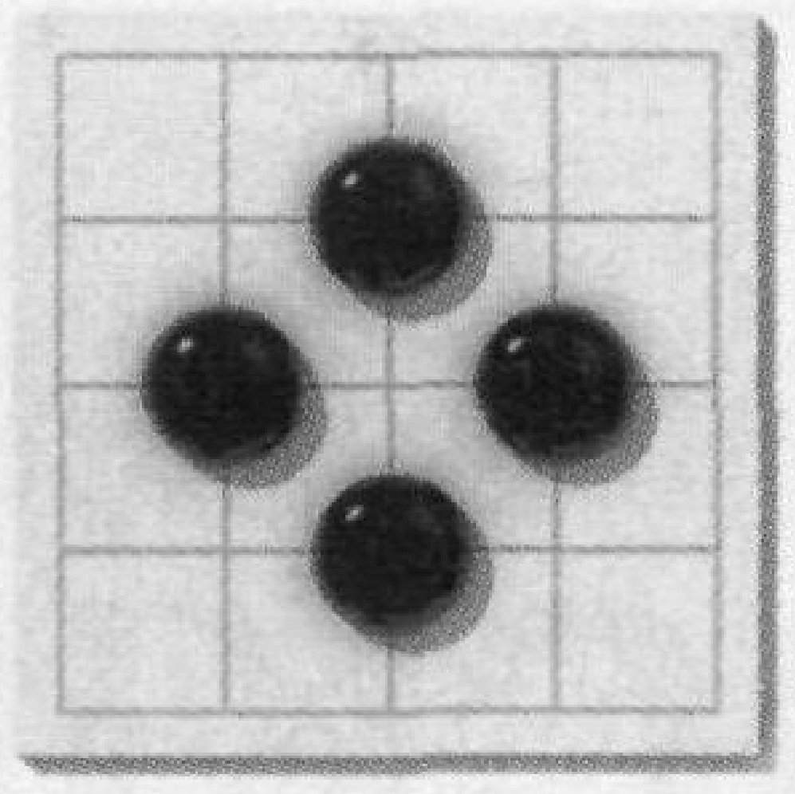
CAPTURING STONES



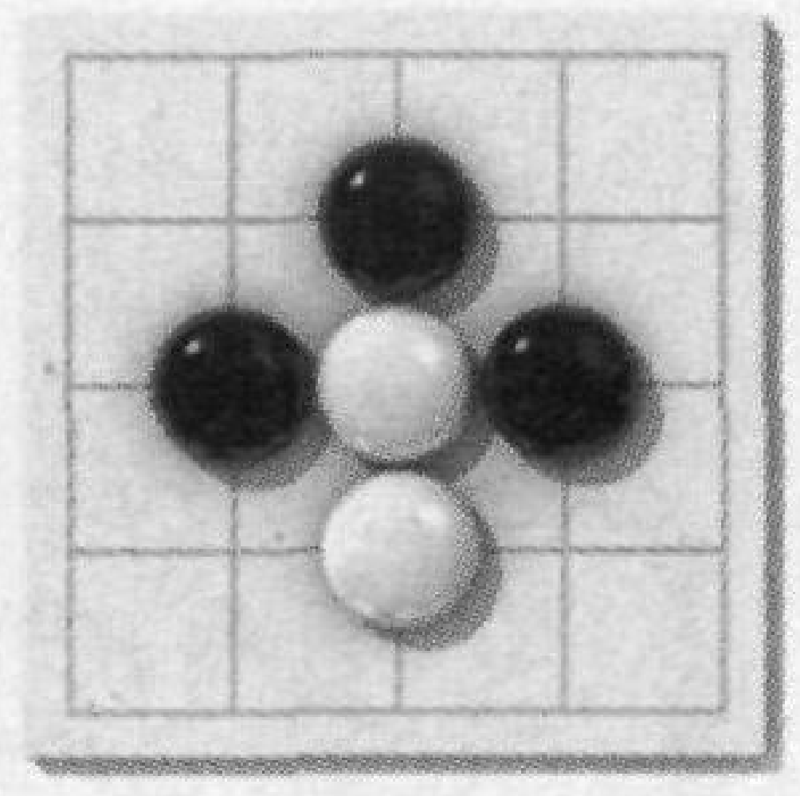
Here, with one more move the white stone will be surrounded...



...leaving it "dead"...



...so that it may be taken off the board.



If it were white's turn to move, however, the player could block the above maneuver, thus.

made progress. The reason is that humans are extremely good at recognizing patterns; it is one of the things that we do best.

It was only in the late 1970s, with the success of Northwestern University's Chess 4.*x* program, written by David Slate and Larry Atkins, that the engineering school of thought became dominant. The idea was to let computers do what *they* do best, namely, calculate. A simple legal-move generator finds all the permissible moves in a position, considers all the possible responses, and then repeats the cycle. Each cycle is called a ply, each generation of new possibilities is called a node—that is, a branching point in a rapidly widening tree of analysis. The branches terminate in "leaf," or end positions.

Carried to its logical extreme, the tree would grow until it exhausted every legal continuation, leaving the program nothing to do but examine the end positions to see which of them were wins—that is, checkmates—and which were draws, then work backward along the branching structure to choose the line that led to the best outcome, assuming that both sides play perfectly. Such exhaustive analysis is impractical, though, because it would produce a tree containing about 10^{60} positions. That's about a thousand times the number of hydrogen atoms in the sun.

There is, however, a course midway between selectivity and exhaustiveness. Instead of analyzing to the end, the program can merely look a few moves further ahead than a human could manage. Deep Blue typically looked 12 plies ahead in all variations (and 40 or more plies in selective lines), generating around 170 million leaf nodes per second. Next, the program would evaluate each of these positions by counting "material," that is, the standard values of the chess pieces. For example, a pawn is worth one point, a knight or bishop three, and so on. Then it added points for a range of positional factors, chosen with the help of human grandmasters.

The resulting evaluation function probably was no better than a middling amateur's ability to grade a single position. But by grading 200 million of them, it was able to do very well indeed. Just ask Kasparov.

This substitution of *search* for *judgment* is the essence of the brute-force method, and it turned out to have two critical advantages over selective search. To begin with, the program became easier to write, had far fewer bugs, and did not have so many blind spots. And crucially, the program played significantly and measurably better as the processing power increased, once the switch to brute force had been made.

Slate and Atkins believed their program was playing at only Class C level—that is, about the level of the typical avid tournament player, who is rated between 1400 and 1600 on the U.S. Chess Federation's rating scale. However, when they moved their program to a supercomputer, it shocked everyone by winning a tournament among Class A players, with ratings between 1800 and 2000. A Class A player is good enough to beat a Class C player 9 times out of 10, on average.

Moving to a supercomputer made this enormous difference because it allowed the program to look just a little further ahead. Detailed measurements later showed that when a brute-force program searched just one ply deeper, its strength improved by between 200 and 300 rating points. When two players are separated by that big a gap, the higher-rated player will win, on average, 4 out of 5 games.

It was this almost linear relationship between search depth and playing strength that first made me believe chess could be solved. I wondered whether the relationship would continue

DIAGRAMS: BRYAN CHRISTIE DESIGN