SCIENTIFIC AMERICAN
JUNE 2007 WWW.SCIAM.COM

COMPUTING

Silicon Smackdown

NEW GO ALGORITHM AIMS TO DEPOSE HUMANS **BY KAREN A. FRENKEL**

A decade ago IBM's chess program, Deep Blue, beat world champion Garry Kasparov in a six-game match. The event marked a milestone, forcing humans to yield dominance of yet another strategic diversion. Only the Asian board game Go seemed to be computer science's Achilles' heel: humans could soundly beat the machines. A new algorithm can now take on strong human players—and win.

Go has proved enormously difficult for computer programmers because of the game's deceptive complexity. The objective of Go is to stake out territory and surround an opponent by placing black or white stones on the intersections of a nine-by-nine or 19-by-19 line grid. Especially on the large board, the number of possible moves per turn is huge—200 on average for each midgame position compared with the several dozen possible in chess. There are also enormous branching factors. Given N positions on the board, the total number of possible game positions is 3^N, because every position can be occupied by a black or white piece, or it can be empty. The total number of legal positions on the small board is about 10^{38}; on the large board, about 10^{170}. Additionally,

more stones do not ensure victory, and players must be able to consider local positions and the board as a whole.

To cope with such an enormous number of options, artificial-intelligence experts have designed algorithms to limit searches, but the programs have not been able to beat the better human players on large boards. Last fall two Hungarian researchers reported that their algorithm outdid the win rates of the best Go programs by 5 percent and could compete with professional Go players on small boards. Levente Kocsis of the Computer and Automation Research Institute at the Hungarian Academy of Sciences in Budapest and Csaba Szepesvári, now at the University of Alberta in Edmonton, developed the algorithm, called UCT (for *u*pper *c*onfidence bounds applied to *t*rees). It extends the well-known Monte Carlo method.

First incorporated into Go programs in the 1970s, Monte Carlo works like a political poll: it performs statistical sampling to predict the behavior or characteristics of a large group. When applied to Go, the algorithm evaluates and ranks candidate moves by playing a large number of random games. But playing the move with the highest score in each position does not guarantee that the player will win the game. Instead this type of search merely restricts the number of relevant potential moves.

UCT takes Monte Carlo further by focusing the search on the most promising moves. "The main idea is to sample actions selectively," Kocsis says. The algorithm must strike a balance, testing alternatives that look the best at the moment to find possible weaknesses and exploring "less optimal-looking alternatives, to ensure that no good alternatives are missed because of early estimation errors," he explains.

UCT calculates indices for moves and selects the move that has the highest index. The algorithm computes the index from the win rate, which describes how often that position leads to a win, as well as the number of times the position has been visited but not played. UCT grows a decision tree in memory and uses it to track these statistics. When UCT encounters a move that has not been visited previously, it adds the move to the tree and plays the rest of the game randomly.

UCT decides if the finished random game is a win or loss, then updates the statistics of all moves made during the game. If the index equals the win rate of the move, the algorithm quickly focuses on the most promising path. But nothing guarantees that an initially successful path will eventually yield a winning move. So when selecting moves, UCT inflates win rates by weighting less visited candidate moves more heavily. The researchers borrowed this idea from bandit problems—selective weighting yields the maximum gain for a gambler playing several slot machines with unknown average payoffs.

Mathematicians Sylvain Gelly of the University of Paris-South and Yizao Wang of the Polytechnic School outside Paris have incorporated UCT into a program they call MoGo. It has a 95 percent better win rate as compared with a previous state-of-the-art Monte-Carlo extension algorithm. Now the top-ranked Go algorithm, MoGo demonstrated its abilities this past spring, vanquishing strong amateur players on nine-by-nine boards and beating weaker ones on large boards. Gelly says that UCT is simple to implement and can be improved. So turning the ultimate corner—ending the reign of professional human Go players—could occur in 10 years, Kocsis states.

GO ATTACK: Humans still rule when it comes to Go, but a new algorithm can topple strong players.

Karen A. Frenkel is based in New York City.

GO BEYOND GAMES

The new Go-playing algorithm, called *upper confidence bounds applied to trees* (UCT), is not limited to games. It applies to any problem that involves choosing the best option, as long as alternatives have an internal treelike structure (that is, a cascading set of choices) and their values can be recursively computed. UCT may prove useful for targeting advertisements on the Web, finding the best settings for an industrial plant or optimizing channel allocation in cellular systems.