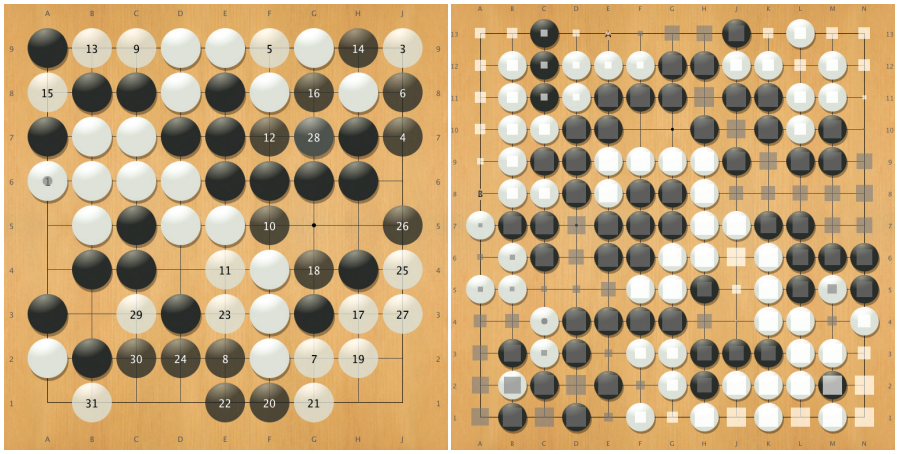
Science | DOI:10.1145/1978542.1978548 Kirk L. Kroeker

A New Benchmark for Artificial Intelligence

Computers are unable to defeat the world's best Go players, but that may change with the application of a new strategy that promises to revolutionize artificial intelligence.

In 1997, when IBM's Deep Blue beat world champion chess player Garry Kasparov in a five-game match, the media heralded the beginning of a new era in artificial intelligence. While the event undeniably marked a noteworthy milestone in the history of computers, and has served as an enduringly fresh metaphor for the possibilities of technology, what became clear in the years following the event is that many classical programming strategies for AI do not work well when applied to more complex applications. One such application that has emerged as a new benchmark for those conducting research in AI is the board game Go.



Go boards showing a sample of how the Fuego software calculates board positions using Monte Carlo Tree Search. The size of the black or white square on each point indicates the percentage of how often that point belongs to each player at the end of a round of simulated moves.

Go has proved to be extremely difficult for computers to master. To date, no computer has beaten a professional Go player on the 19×19 board in an even match. On the surface, Go might appear to be much simpler than chess, with players alternating placement of black and white pieces on a square board to capture more territory than their opponent by game's end. But the simplicity of Go's rules are deceiving. From a computer's perspective, Go is much more complex than chess. Instead of dozens of branching movement combinations to evaluate as in chess, the branching search trees for Go may consist of hundreds of options for each move. For the first two Go moves alone, for example, more than 100,000 lines of play are possible, making options for each player's turn highly open-ended.

The approach that has proved in recent years to be the most likely path to victory for computers in Go is a method called Monte Carlo Tree Search, or MCTS. Only a few years ago, computers couldn't compete well even with amateur Go players, but the use of MCTS has resulted in software that can play near the level of the best professional players on the 9×9 Go board, and can even provide a decent game for seasoned players on the 19×19 board. The demonstrated strength of MCTS applied to Go has drawn attention from many areas of computer science, with

IMAGES COURTESY OF MARTIN MÜLLER, UNIVERSITY OF ALBERTA

The Huygens supercomputer running MoGo software, which claimed the first nine-stone handicap victory against a professional Go player on a 19x19 board. During the official match, the supercomputer used 800 processors to power MoGo at a speed approximately 1,000 times faster than IBM's Deep Blue.

researchers now showing interest in applying MCTS techniques to other applications and domains.

Brute force methods, such as the classic alpha-beta search technique, are far too slow for Go analysis because of the game's open-ended nature. But with an MCTS algorithm, essentially a form of statistical sampling, it is possible to quickly consider all possible moves on the board, then simulate a set of random games. If a winning combination turns up in the majority of those simulated games, the MCTS algorithm considers that combination a good one. Otherwise, the algorithm continues to search. Currently, the most effective MCTS algorithms balance lengthy deep searches for winning moves against simulated explorations of new positions.

Currently, the most effective MCTS algorithms balance lengthy deep searches for winning Go moves against simulated explorations of new positions.

One scientist doing work to improve the capabilities of MCTS is Martin Müller, a professor in the department of computer science at the University of Alberta. Müller is co-author of Fuego, a Go program that routinely places in the top ranks at the annual Computer Games Olympiad and was the first program to beat a professional Go player in an official game on the 9×9 board. Müller has worked on Go for more than 20 years, dating back to his Ph.D. dissertation at ETH Zürich, where he used combinatorial game theory to solve endgame problems.

Now using MCTS, he says the method promises to push the boundaries of what is possible in AI. The Fuego MCTS framework, for example, has already been used in many other applications. "As a long time computer Go researcher and Go player, I enjoy gaining new insights from the program's play as it becomes stronger than I am," he says. "This has happened on the 9×9 board, where Fuego has achieved some victories against top-level human competition." But the ultimate challenge, he says, is to scale the program so that it can beat humans on the 19×19 board.

MCTS's Superior Approach

Müller points out that while it has been somewhat of a mystery why MCTS has proved superior to earlier approaches, researchers are beginning to understand why MCTS methods can be more successful in hard domains. "For me, the main reason is that they do not require an evaluation function for arbitrary game positions," he says. "The simulations until the end of the game in MCTS make evaluating the best possible positions much easier."

Even so, the best Go programs are still weak in what Müller calls local situations. Massive parallelization is being used to sidestep this problem, but Müller says it is currently unrealistic to expect supercomputers to be able to resolve a large number of local situations simultaneously within a single global search. "Simply scaling to more and faster processors will not be enough with current techniques," he says. "I think we need one or two further breakthrough ideas in algorithms."

That breakthrough, Müller suggests, might come from MCTS being combined with other methods. For example, most strong algorithms are inherently sequential, meaning the results of all simulations must be derived before determining which simulations are run next. MCTS algorithms can sidestep this process with a technique called virtual loss, which operates on the assumption that simulations that have been started in parallel, but have not yet completed, will lead to a loss. While this technique is fast because it does not search the same simulations repeatedly, it is less effective than sequential search to resolve local situations, and it reaches a performance ceiling, even when parallelized.

What might lead to solving this problem, says Müller, is a method that can supplement global search with a set of effective local searches. "I believe we need local MCTS as well, and to integrate the results with global search," says Müller. "This is a huge technical challenge, but if successful, it would provide a way to scale up much better."

As for applying MCTS to other domains, Müller says the main challenge is to overcome the method's known limitations, particularly in cases where promising states are sparse, making them difficult to discover, or where

IMAGES COURTESY OF INRIA/CNRS/LRI (FRANCE), NUTN (TAIWAN), AND SARA (NETHERLANDS)

planning problems have extremely long solutions. In these cases, hybrid approaches, such as limited-length simulations followed by classical evaluations, might turn out to be the most effective strategies. "We will need to understand how classical techniques and MCTS can best be combined," he says.

Another researcher working on applying MCTS to Go and other domains is Olivier Teytaud, a computer scientist at the University of Paris-Sud, and coauthor of MoGo, a Go program that has won several significant victories against human players and other Go programs. Teytaud's early work focused on planning for industrial problems, but he shifted his attention to Go and MCTS several years ago. Like Müller, Teytaud is dedicated to improving the capabilities of MCTS in Go, but with an eye toward applying the ideas to other applications and domains.

"It's clear that curing cancer, reducing pollution, or automating tedious tasks are more important than playing the game of Go," he says. "But games are a very clear challenge, providing a great test bed for comparing algorithms."

Teytaud says one of the reasons he became interested in MCTS is because of its ability to bypass the extrapolation problem. Unlike classical approaches that attempt to generalize domain knowledge to new scenarios, MCTS does not attempt to extrapolate. Instead, MCTS relies on search-tree simulations that require minimal domain knowledge, making it an attractive option for problems in AI. However, like Müller, Teytaud is focused on how best to get MCTS methods to handle local situations that would benefit from the kind of abstraction that humans can do so well.

"We have clearly understood the weaknesses of MCTS, and we are trying many different things to solve them," Teytaud says. "But we don't currently know which direction is the best to take."

Progress in Other Fields

Despite the current mysteries surrounding MCTS, and no clear way to overcome the evident weaknesses of the approach, Teytaud says some progress has been made in applying the method to other applications in which extrapolation is difficult. "Power-plant management is my favorite application because of its strong ecological and economic importance," he says. "We would never have been given funding for working on power-plant management without the publicity provided by the work on Go."

The main challenge of applying MCTS to other domains is to overcome the method's known limitations, particularly in cases where promising states are sparse, making them difficult to discover, or where planning problems have extremely long solutions, says Martin Müller.

In his most recent effort along these lines, Teytaud is working directly with Artelys, a company that specializes in power-plant management, to develop a system that can respond intelligently to changes in power demand and outages. Managing power allocation, says Teytaud, is a problem similar to Go, where an opponent's move may be likened to the failure of a plant, the computer's response likened to switching other plants on or off, and the territory occupied at the end of a Go match likened to the ecological or economic benefit derived from proper power allocation.

Besides being useful for industrial applications, MCTS techniques developed for Go appear to be well suited to multiplayer games. An MCTS-based AI application that Teytaud developed for the massively multiplayer card game UrbanRivals was ranked in the top 1% in a round of matches consisting of some 9,000 players. "The success of this application illustrates that the MCTS method can deal with partially hidden information," he says. "There's no hidden information in Go, so this application to card games is important."

As for whether a computer will be able to achieve victory over a top-ranked Go player in an even match, Teytaud says he remains skeptical about it happening soon. "It will probably include some abstract thinking, much more than for chess," he says. Like Teytaud, Müller says he is uncertain whether or when the technique will be used to beat a professional Go player on the 19×19 board. "I have no crystal ball," he says, "but I hope to see at least one more fundamental breakthrough in the next decade."

Even if a landmark victory—on the order of Deep Blue over Kasparov—is not on the horizon for MCTS and the game of Go, the popularity of MCTS likely will continue to grow in domains suited to heuristic search methods, such as industrial-optimization problems, multiplayer games, and other applications that stand to benefit from more effective AI strategies. C

Further Reading

Arneson, B., Hayward, B., and Henderson, P. **Monte Carlo tree search in hex, *IEEE Transactions on Computational Intelligence and AI in Games 2*, 4, Dec. 2010.**

Chaslot, G.M., Winands, M.H., and Herik, H.J. **Parallel Monte-Carlo tree search, *Proceedings of the Sixth International Conference on Computers and Games*, Beijing, China, Sept. 29–Oct. 1, 2008.**

Enzenberger, M., Müller, M., Arneson, B., and Segal, R. **Fuego: An open-source framework for board games and Go engine based on Monte-Carlo tree search, *IEEE Transactions on Computational Intelligence and AI in Games 2*, 4, Dec. 2010.**

Rimmel, A., Teytaud, O., Lee, C.S., Yen, S.J., Wang, M.H., and Tsai, S.R. **Current frontiers in computer Go, *IEEE Transactions on Computational Intelligence and AI in Games 2*, 4, Dec. 2010.**

Winands, H.M., Bjornsson, Y., and Saito, J. **Monte Carlo tree search in lines of action, *IEEE Transactions on Computational Intelligence and AI in Games 2*, 4, Dec. 2010.**

Based in Los Angeles, **Kirk L. Kroeker** is a freelance editor and writer specializing in science and technology.