About

What is MCTS?

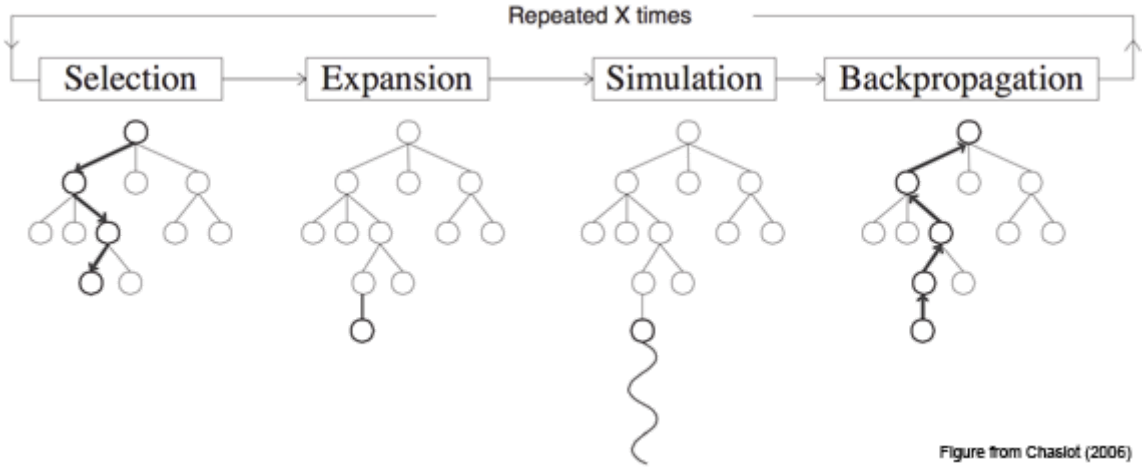
Monte Carlo Tree Search (MCTS) is a method for making optimal decisions in artificial intelligence (AI) problems, typically move planning in combinatorial games. MCTS combines the generality of random simulation with the precision of tree search.

John von Neumann's 1928 minimax theorem paved the way for adversarial tree search methods that have formed the basis of decision making in computer science and AI almost since their inception. Monte Carlo methods were later formalised in the 1940s as a way to approach less well-defined problems unsuitable for tree search through the use of random sampling. Rémi Coulomb combined these two ideas in 2006 to provide a new approach for move planning in Go now known as MCTS.

Research interest in MCTS has risen sharply due to its spectacular success with computer Go and potential application to a number of other difficult problems. Its application extends beyond games, and MCTS can theoretically be applied to any domain that can be described in terms of {*state*, *action*} pairs and simulation used to forecast outcomes.

Basic Algorithm

The basic MCTS algorithm is simplicity itself: a search tree is built, node by node, according to the outcomes of simulated playouts. The process can be broken down into the following steps:



1. Selection
Starting at root node *R*, recursively select optimal child nodes (explained below) until a leaf node *L* is reached.

2. Expansion
If *L* is a not a terminal node (i.e. it does not end the game) then create one or more child nodes and select one *C*.

3. Simulation
Run a simulated playout from *C* until a result is achieved.

4. Backpropagation
Update the current move sequence with the simulation result.

Each node must contain two important pieces of information: an estimated value based on simulation results and the number of times it has been visited.

In its simplest and most memory efficient implementation, MCTS will add one child node per iteration. Note, however, that it may be beneficial to add more than one child node per iteration depending on the application.

Node Selection

Bandits and UCB

Node selection during recursive tree descent is achieved by choosing the node that maximises some quantity, analagous to the *multiarmed bandit problem* in which a player must choose the slot machine (bandit) that maximises their estimated reward each turn. The following Upper Confidence Bounds (UCB) formula is typically used:

$$v_i + C \times \sqrt{\frac{\ln N}{n_i}}$$

where v_i is the estimated value of the node, n_i is the number of the times the node has been visited and N is the total number of times that its parent has been visited. C is a tunable bias parameter.

Exploitation vs Exploration

The UCB formula provides a good balance between the *exploitation* of known rewards and the *exploration* of relatively unvisited nodes to encourage their exercise. Reward estimates are based on random simulations, so nodes must be visited a number of times before these estimates become reliable; MCTS estimates will typically be unreliable at the start of a search but converge to reliable estimates given sufficient time and perfect estimates given infinite time.

MCTS and UCT

Kocsis and Szepervari (2006) first formalised a complete MCTS algorithm using UCB and dubbed it the Upper Confidence Bounds for Trees (UCT) method. This is the algorithm used in the vast majority of current MCTS implementations.

UCT may be described as a special case of MCTS, that is:

UCT = MCTS + UCB

Benefits

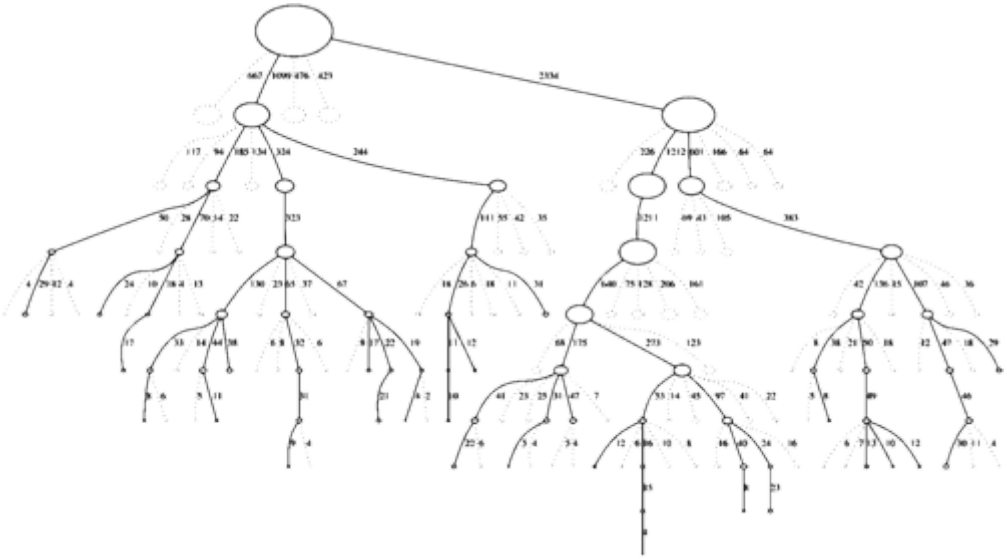
MCTS offers a number of advantages over traditional tree search methods.

Contextual Freedom

The greatest benefit of MCTS is that it does not require any strategic or tactical knowledge about a given game (or other problem domain) to make reasonable move decisions. The algorithm can function effectively with no knowledge of a game apart from its legal moves and end conditions; this means that a single MCTS implementation can be reused for any number of games with little modificiation, and makes MCTS a potential boon for general game playing.

Asymmetric Tree Growth

MCTS performs an asymmetric tree growth that adapts to the topology of the search space. The algorithm visits more interesting nodes more often, and focusses its search time in more relevant parts of the tree.



This makes MCTS suitable for games with large branching factors such as 19x19 Go. Such large combinatorial spaces typically cause problems for standard depth- or breadth-based search methods, but the adaptive nature of

MCTS means that it will (eventually) find those moves that appear optimal and focus its search effort there.

Graceful Exit
The algorithm can be halted at any time to return the current best estimate. The search tree built thus far may be discarded or preserved for future reuse.

Simplicity
The algorithm is extremely simple to implement (see the Tutorial).

Drawbacks

MCTS has few drawbacks, but they can be major.

Playing Strength
The MCTS algorithm, in its basic form, can fail to find reasonable moves for even games of medium complexity within a reasonable amount of time. This is mostly due to the sheer size of the combinatorial move space and the fact that key nodes may not be visited enough times to give reliable estimates.

Luckily, the performance of the algorithm can be sigificantly improved using a number of techniques.

Improvements

There are two types of improvements that may benefit an MCTS implementation: those specific to the current domain and those common to all domains.

Domain Specific

Domain knowledge specific to the current game is typically exploited during the simulation stage to produce playouts that are more similar to playouts that would occur between human opponents. This means that playout results will be more realistic than random simulations and that nodes will require fewer iterations to yield realistic reward values.

Domain specific improvements typically involve the completion of local move patterns known to work for the current game, such as capturing moves in Go or bridge intrusions in Hex. They can yield significant performance improvements for the current game at the expense of generality.

Domain Independent

Domain independent enhancements with broad application to a range of applications are the holy grail of MCTS research, and are the focus of most current work in the area. Dozens of such enhancements have been suggested to date with varying degrees of success, ranging from the simple (play winning moves/avoid losing moves where possible during playouts) to more complex node initialisation and selection methods and meta-strategies.

See the Enhancements page for a more detailed list of MCTS improvements.

Open Research Topics

As MCTS is a new field of study, there are many open research topics.

Algorithmic Enhancements

Almost all of the dozens of improvements suggested for the basic algorithm require further research. See the list in the Enhancements section for more details.

Automated Parameter Tuning

One of the simplest open questions is how to dynamically adjust search parameters such as the UCB bias term to maximum effect, and what other aspects of the search may be similarly parameterised.

Node Expansion

Some applications benefit from expanding the search tree one node per iteration whereas other might benefit from expanding all possible nodes per iteration. There are to date no clear guidelines as to which strategy should be applied for a given situation, and whether this can be determined automatically.

Node Reliability

It would be useful to know how many times a given node must be visited before its reward estimate is sufficently reliable, based on its situation and relative position within the search tree.

Tree Shape Analysis

We have already conducted preliminary work into the question of whether UCT tree shape might yield some insight into the characteristics of a given game (Williams 2010).The results are encouraging.