4 Variations

Wednesday, February 5, 2020 8:48 PM

4.1 Flat UCB (Coquelin and Munos)

Flat UCB effectively treats the leaves of the search tree as a single multi-armed bandit
The adaptivity of standard UCT while improving its regret bounds in certain worst case
optimistic.

4.2 Bandit Algorithm for Smooth Trees (BAST) (Coquelin and Munos)

BAST ignores branches that are suboptimal with high confidence. They applied BAST to
approximation and showed that only branches that are expanded indefinitely are the c
contrast, plain UCT expands all branches indefinitely.

4.3 Learning in MCTS

MCTS can be seen as a type of Reinforcement Learning algorithm. It is interesting to co
with temporal difference learning

4.3.1 Temporal Difference Learning (TDL) (Silver et al.)

TDL is equivalent to MCTS when the state values can be stored directly in a table. MCTS
state values in order to decide the next move, whereas TDL learns the long-term value
guides future behaviors. TDL can learn heuristic value functions to inform the tree polic
policy.

4.3.2 Temporal Difference with Monte Carlo (TDMC) (Osaki et al.)

Osaki described the Temporal Difference with Monte Carlo algorithm as a new method
learning using winning probability as substitute rewards in non-terminal positions and
performance over standard TD learning.

problem. UCB retains
s where UCT is overly

Lipschitz function
optimal ones. On

nsider its relationship

S estimates temporary
of each state that then
cy or the simulation

l of reinforcement
report superior

4.3.3 Bandit-Based Active Learner (BAAL) (Rolet)

BBAL is proposed to address the issue of small training sets in applications where data
Active learning is formalized under bounded resources as a finite-horizon RL problem v
minimizing the generalization error.

The optimal policy is approximated by a combination of UCT and billiard algorithms.
Progressive widening is employed to limit the degree of exploration by UCB1 to give pr
results.

4.4 Single-Player MCTS (SP-MCTS)

It adds a third term to standard UCB formula that represents the possible deviation of

$$\sqrt{\sigma^2 + \frac{D}{n_i}}$$

Where σ^2 is the variance of the node's simulation result, and D is a constant. $\frac{D}{n_i}$ can inf
deviation for infrequently visited nodes, so that the rewards for such nodes are consid
SP-MCTS can also uses a heuristically guided default policy for simulations.

4.4.1 Feature UCT Selection (FUSE)

FUSE uses UCB1-Tuned and RAVE to choose the stopping feature during simulation acc
the tree.

4.5 Multi-player MCTS

mAx^n *idea is* used to apply MCTS to multi-player games: each node stores a vector
selection procedure seeks to maximize the UCB value calculated using the approxi
reward vector.

simulation according to the depth of the tree.

4.5 Multi-player MCTS

mAx^n *idea is* used to apply MCTS to multi-player games: each node stores vector of rewards, and the selection procedure seeks to maximize the UCB value calculated using the approximate component of the reward vector.

Players acting in coalition:

- Paranoid UCT: the player considers that all other players are in coalition against him
- UCT with Alliances, the coalitions are provided explicitly to the alorithm
- Coffident UCT: indepent searches are conducted for each possible coalition of the searching player with one other player, and move choosen according to whichever of these coalitions appear to be most favorable.

4.5.1 Coalition Reduction

Multiple cooperative oppents can be reduced to a single effective opponent

4.6 Multi-agent MCTS

The simulation phase of UCT is a single agent playing against itself. The emergent properies of interactions between different agent types would lead to exploration of the search space. However, finding the set of agents with the correct properties (i.e thoes that increase playing strength) is computationally intensive.

4.6.1 Ensemble UCT

Multiple instsances of UCT are run independently and their root statistics combined to yield the final result. This approach is closely related to root parallelisation and also to deteminization.

Chaslot argues that. For Go, Ensemble UCT with n instances of m iterations outperforms plain UCT with mn iterations. But he is not able to provide evidence on other domains

4.7 Real-time MCTS

Simulation-based (anytime) algorithms such as MCTS are well suited to domains in which time per move is limited. Approximators may be used to increase the efficency of the forward model.

4.8 Nondeterministic MCTS

Traditional AI game research typically focuses on deterministic games with perfect information.
Opponent modelling is very important in games of imperfect information games.

4.8.1 Determinization

A stochastic game with imperfect information be transformed into a deterministic game with perfect information, by fixing the outcomes of all chance events and making states fully observable.

Determinization is the process of sampling several instances of the deterministic game with perfect information, analyzing each with standard AI techinques, and combining those analysis to obtain a decision for the full game.

4.8.2 Hindsight Optimization (HOP)

The idea is to obtain an upper bound on the expected reward for each move by assuming the ability to optimize one's subsequent strategy with "hindsight" knowledge of future outcomes.
The upper bound can be approximated by determinization.

4.8.3 Sparse UCT

Sparse UCT is the generalization of HOP-UCT procedure. In sparse UCT, a node may have several children corresponding to a different stochastic outcome of that move, but traversal to child nodes and expansion are stochastic.

4.8.4 Information Set UCT (ISUCT)

Strategy fusion is a problem with determinization techniques, which involves the incorrect assumption that different moves can be chosen from different states in the same information set.

All information sets are from the point of view of the root player. Each iteration samples a determinization and restricts selection, expansion and simulation to thoes parts of the tree compatible with the deteminiation. The UCB formula is modified to replace the "parent visit" count with the number of parent visits in which the child was compatible.

ISUCT fails to outperform determinized UCT overall. However, ISUCT is shown to perform well in precisely thoes situations where access to hidden information would have the greatest effect on the outcome of the game.

4.8.5 Multiple MCTS (MMTCTS)

It is a variant of MCTS for games of imperfect information, in which multiple trees are searched simultaneously. Specifically, there is a tree for each player, and search descends and updates for call the trees simulatnenously, using statstics in the tree for the relevant player at each stage of selection.

4.8.6 UCT+

Maximize

$$\overline{X}_j + c\sigma_{\overline{X}_j}$$

where $\overline{X}_j$ is the average reward from action j, $\sigma_{\overline{X}_j}$ is the standard error on $\overline{X}_j$, and c is a constant.

At the oppnent nodes are treated as chance nodes with probabilities determined by an opponent model.
During backpropagation, the calculations are weighted by the probabilities of the actions leading to each child.

4.8.7 Monte Carlo **α**-β (MCab)

Mcab combines MCTS with traditional tree search by replacing the default policy with a shallow a-b search.

4.8.8 Monte Carlo Counterfactual Regret (MCCFR)

CFR is an algorithm for computing approximate Nash equilibraia for games of imperfect information.

MCCFR works by sampling blocks of terminal histories (paths through the game tree from root to leaf) and computing immdiate counterfactual regrets over these blocks

4.8.9 Inference and Oppoent Modelling

In a game of imperfect information, it is possible to infer hidden information from opponent actions.

The opponent model has two parts

1. A prior model of a general opponent
2. A corrective function for the specific opponent
   Which are learnt from samples of previously played games

4.8.10 Simultaneous Moves

Simultaneous moves can be considered as a special case of hidden information: one player chooses a move and hides it, then the other player chooses a move and both are revealed.

4.9 Recursive Approaches

The following methods recursively apply a MC technique to grow the search tree. These have had success with single-player puzzles and similar optimization tasks.

4.9.1 Reflexive Monte Carlo Search

Each layer simply informed by the one below

4.9.2 Nested Monte Carlo Search (NMCS)

Memorizing the best sequence of moves found at each level of the search so far

4.9.3 Nested Rollout Policy Adaptation (NRPA)

Is an extension of nested Monte Carlo search in which a domain-specific policy is associated with the action leading to each child

4.9.4 Meta-MCTS

It replaces the default policy with a nested MCTS program that plays a sub-game in their Meta-MCTS algorithm. Meta-MCTS: Quasi Best-First or Beta Distribution Sampling

4.9.5 Heuristically Guided Swarm Tree Search

breadth-first search

4.10 Sample-Based Planners