Latex Sample File

James Parker
jparker@cs.umn.edu

April 27, 2016

1 Introduction

In this article, we explore the various implementations of Monte Carlo Simulations and their impact within the environment they are used. Some of these implementations only change at a structure level, while others change at a high level to fit the needs of a specific problem. These implementations include Model Based Monte Carlo Search for Playing Atari Games, Parallelized Monte Carlo Tree Search, Monte Carlo Planning for Real Time Strategy Games, [insert kohei topics]

2 The Monte Carlo Tree Search Idea

The basis of Monte Carlo Simulation came out of the limitations of other Artificial Intelligence. Classic approaches from the past relied on on a game dependent heuristic to give an AI an estimate of the current game state. However, making making mid state evaluators that reliably evaluated heuristic values are too complex and depend too much on the nature of the game. (Monte-Carlo Tree Search: A New Framework for Game AI) Monte Carlo Tree Search does not have this problem because it does not rely on heuristics to get a guess of the game state. Rather, it builds simulations of the game and then builds statistics to choose the next action with the highest win rate of the simulations. This is done using four stages: Selection finds a state in the tree and decides based on the statistics aggregated which node to explore. There is a balance that needs to be achieved such that it does not keep picking a local optimum to allow the exploration of a possible better game state. Expansion is when the game reaches a node that is currently not in the search tree, and a new node is added. From there, Simulation takes over, making random moves for both the parties. Backpropagation happens when the simulation reaches a terminal node, and the result of that simulation is sent upwards throughout the path that the selection phase took, updating the win rates of those nodes.

3 Simulation Based Approach to General Game Playing

Because of nature of Monte Carlo Simulations not being reliant on game dependent mid-state evaluators to obtain heuristics, this approach can be implemented in various games. A modification to Monte Carlo Search Trees simulation phase, Upper Confidence-bounds applied to Trees or UTC for short, is introduced in a General Game Playing Competition and won in 2007. (Simulation Based Approach to General Game Playing) This approach modifies the Selection aspect such that it gives a good balance between exploration and exploitation of nodes. To do this, not only is the win rate of each possible action at the current game state kept track, but also the amount of times that action is visited. This allows the AI to prioritize its simulations elsewhere if the action has already been simulated multiple times, allowing for the discovery of better choices in a global sense rather than just focusing on the local best choice, and balances it with the limited time and memory that the AI has to simulate and retrieve statistics for each possible move. GGP consists of a variety of games, each with their own level of computational difficulty, and thus it is important that the algorithm be able to prioritize its selection of action nodes properly such that it wont need new hardware or more time.

4 Monte Carlo Planning in RTS Games

While the assortment of games may present Monte Carlo a good approach for game AIs it still only covered games that are for the most part, turn based. In Real Time Strategy, there is no turn based action, and all actions of players can be done regardless of each other. Many of the supposed AIs for RTSs are mainly scripts, resulting in players wanting to play against other human players since the AIs are considered beginner level. An implementation of Monte Carlo idea, Monte Carlo Planning can handle RTS. It follows the same approach as Monte Carlo Tree search, but instead of searching nodes, it searches plans that are randomly generated and then evaluated via repeated simulation and backtracking. Each plan is one of three types of plans: unit control, combat planning, and strategic planning. These plans are randomly generated and simulated forwards to see which one gives the best win rate. The benefit of using this is that it does not have to rely on perfect information. Most AI need to rely on either the heuristic to guess the optimal solution, or have perfect information of the game state to evaluate, which is extremely hard in multiplayer RTS games, sometimes even impossible. Planning via Monte Carlo does not have to rely on that because it relies on statistics and randomness to sample the game state for the best plans.

5 Model Based Reinforcement Learning for Playing Atari Games

One of the challenges of using the Monte Carlo Search Idea is that the Selection phase can become computationally expensive if it keeps selecting nodes that lead to regions of the search tree that doesn't look promising. Another way of balancing exploration and exploitation is to use weighted Monte Carlo Tree Search. This AI gives each action node from the current state a probability of being picked in selection based on its current win rate and then a node is randomly picked. In contrast to uniform random selection or select highest win rate node, this allows for the possibility of picking a low win node and possibly discovering a better action while still favoring nodes with better win rates, making the algorithm as a whole more reactive. In Model Based Reinforcement Learning for Playing Atari Games, the game of pong was used to evaluate the effectiveness of normal Monte Carlo Tree Search and weighted Monte Carlo, the weighted version was found to react faster to the ball than uniform random Monte Carlo. At long distances, the paddle wont even move to the general vicinity of the ball because the goal state is too deep in the search tree to be found. In weighted, this is mitigated by giving a majority of simulation time to more promising actions.

6 Parallel Monte Carlo Tree Search

With the advent of new hardware paradigms, implementations of Monte Carlo can take advantage of these paradigms. One paradigm that Monte Carlo is set to take advantage of is parallelism in hardware. From this, four implementations Monte Carlo Search have arisen. Of the four, the two that give insight are Leaf and Root Parallelization (Parallel Monte Carlo Tree Search). Leaf is one way to apply parallelism to the simulation phase of MCTS. That is, multiple threads are assigned the task of carrying out simulations. This allows for multiple samples on one selection phase, giving a more accurate distribution of win rates across the search nodes. Root on the other hand seeks to parallelize from the root up. This means that full phase of MCTS are parallelized and the grand total of each search nodes are totaled up and the best action is selected based on that.

7 Conclusion

As the field of AI increases, the prevalence of Monte Carlo implementations are increasing.. All versions, while having their unique qualities that improve its performance, still are for the most part are the same in their philosophy. Such a powerful idea that it can be implemented in most situations where other AIs would fail show that Monte Carlo method is the preferred method for games that are complex and dynamic.

References

[1] Carlos Hernández, Roberto Asín, and Jorge A Baier. Reusing previously found A* paths for fast goal-directed navigation in dynamic terrain. In *AAAI*, pages 1158–1164, 2015.