Monte Carlo Tree Search

Mark Maloof
Department of Computer Science
Georgetown University
Washington, DC 20057

1 January 1970

Overview

- MCTS consists of four main steps (Browne et al., 2012)
  1. Selection: Starting at the root, select the best action until reaching a node that has not been fully explored (i.e., a node with untried and therefore unevaluated actions).
  2. Expansion: Choose an action, and expand the tree by adding a child node.
  3. Simulation: From the newly added child, uniformly randomly select actions until reaching a leaf node and receiving a reward (e.g., $+1$ for winning, -1 for losing).
  4. Backpropagation: Starting at the new child node, propagate the reward to the root by adjusting the visit count $N(v)$ and the simulation reward $Q(v)$ of the nodes along the path.

Figure 2, Brown et al. (2012)

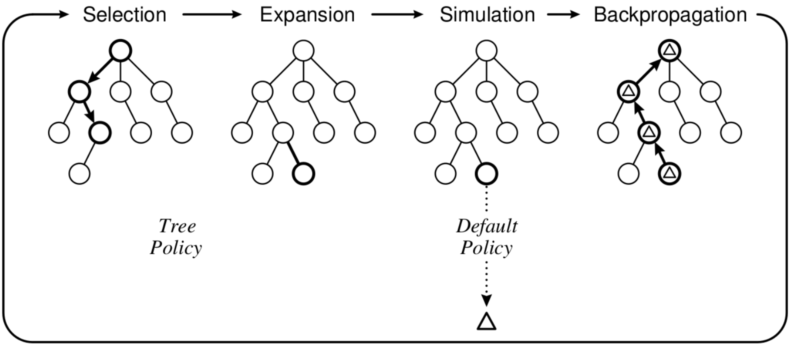

Fig. 2. One iteration of the general MCTS approach.

Upper-confidence Bound for Trees (UCT)

function uctSearch(s_0)
    create a root node v_0 with state s_0
    while within computational budget **do**
        $v_l \leftarrow$ treePolicy(v_0)
        $\Delta \leftarrow$ defaultPolicy(($s(v_l)$))
        backup(v_l, Δ)
    end while
    return a(bestChild($v_0, 0$))
end function

Tree Policy

```
function treePolicy(v)
    while v is non-terminal do
        if v not fully expanded then
            return expand(v)
        else
            v ← bestChild(v, C_p)
        end if
    end while
    return v
end function
```

Expand

```
function expand(v)
    choose a ∈ untried actions from A(s(v))
    add a new child v′ to v with s(v′) = f(s(v), a) and a(v′) = a
    return v′
end function
```

Best Child

```
function bestChild(v, c)
    return argmax_{v' ∈ Children(v)} Q(v')/N(v') + c sqrt(2 ln N(v) / N(v'))
end function
```

$$\text{return } \operatorname{argmax}_{v' \in \mathrm{Children}(v)} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2\ln N(v)}{N(v')}}$$

Default Policy

```
function defaultPolicy(s)
    while s is non-terminal do
        choose a ∈ A(s) uniformly at random
        s ← f(s, a)
    end while
    return reward for state s
end function
```

Backup

function backup(v, Δ)
    while s is not null **do**
        $N(v) \leftarrow N(v) + 1$
        $Q(v) \leftarrow Q(v) + \Delta(v, p)$ $\triangleright$ p is player
        $v \leftarrow$ parent of v
    end while
end function

Backup Negamax

function backupNegamax(v, Δ)
    while s is not null **do**
        $N(v) \leftarrow N(v) + 1$
        $Q(v) \leftarrow Q(v) + \Delta$
        $\Delta \leftarrow -\Delta$
        $v \leftarrow$ parent of v
    end while
end function

Figure 3, Brown et al. (2012)

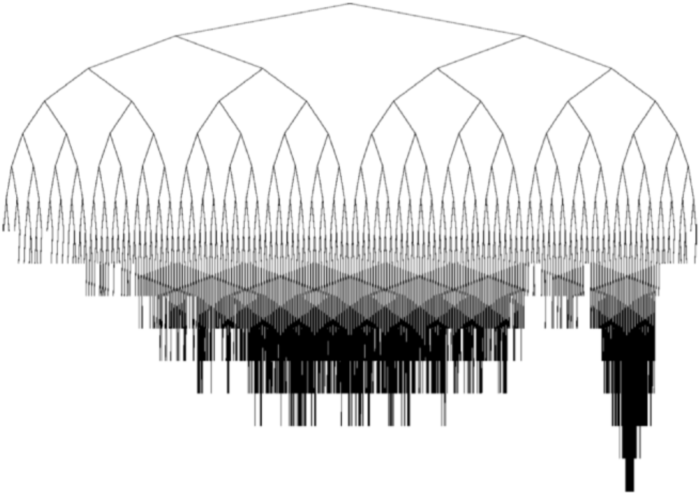

Fig. 3. Asymmetric tree growth [68].

Monte Carlo Tree Search

Mark Maloof
Department of Computer Science
Georgetown University
Washington, DC 20057

1 January 1970

References I

C. Browne, E. Powley, D. Whitehouse, S. Lucas, P. I. Cowling, P. Fohlfshagen, S. Tavener, D. Perez, S. Samothrakis, and S. Colton. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.