Monte Carlo Tree Search in MDPs

20. května 2019

B4M36PUI/BE4M36PUI — Planning for Artificial Intelligence

Outline

- Review of last tutorial
- Upper Confidence Bound for Trees
- Assignment Q&A

Review of previous tutorial

Stopping criterion for prioritized VI

Question: What is the stopping criterion for prioritized VI?

Question: What is the convergence criterion for prioritized VI?

Stopping criterion for prioritized VI

Question: What is the stopping criterion for prioritized VI?

Question: What is the convergence criterion for prioritized VI?

Algorithm 2: Prioritized VI

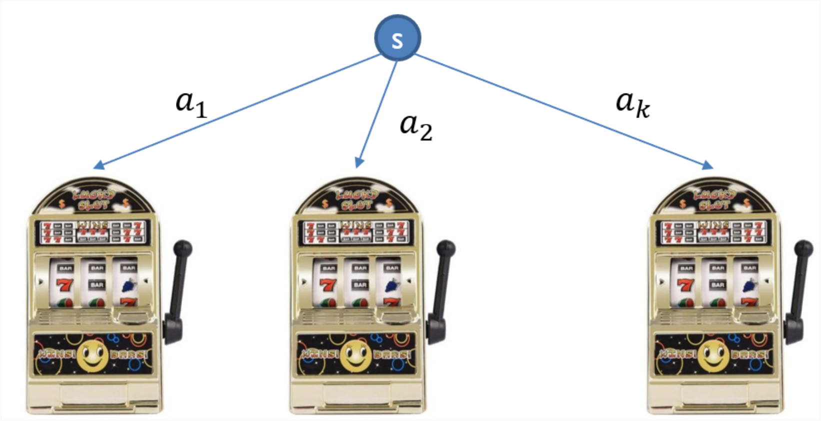
```
1 initialize $V_0$ e.g to 0 and initialize priority queue $q$
2 while $Res^V > \epsilon$ do
3     pick state $s'$ according to priority: $s' = q.pop()$
4     Bellman backup on $s'$: $V(s') \leftarrow \max_{a \in A} \sum_{s \in S} T(s', a, s)[R(s', a, s) + \gamma V(s)]$
5     Update residual at $s'$: $Res^V(s') = |V_{\text{old}}(s') - V_{\text{new}}(s')|$
6     foreach $s$ predecessor of $s'$, i.e. $\{s | T(s, a, s') > 0$ for some $a\}$ do
7         Update priority of $s$:
          $\text{priority}_{PS}(s) \leftarrow \max\{\text{priority}_{PS}(s), \max_{a \in A}\{T(s, a, s') Res^V(s')\}\}$
8 return greedy policy $\pi^V$
```

UCT

K-armed bandit problem

- Each bandit has different *mean* reward

Question: Given M pulls, how do you choose which action (arm) to pull?



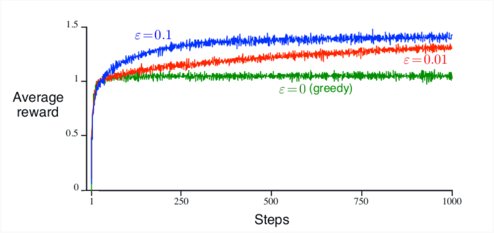
K-armed bandit problem

Question: Given M pulls, how do you choose which action (arm) to pull?

Expected value of each arm: $Q^*(a) = E(R_t|\pi(t) = a)$

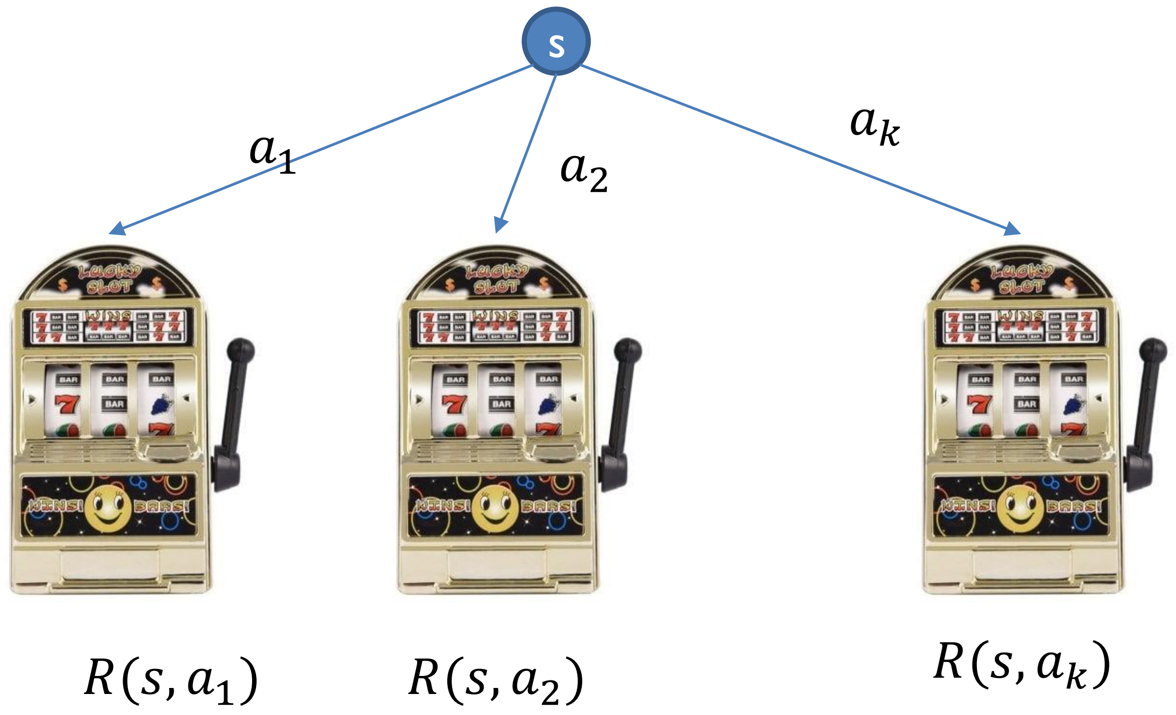
Empirical mean of each arm at time n, after $n_j = \sum_{i=0}^{n} 1_{\pi(i)=a_j}$ pulls on $j - th$ arm: $Q_n(a_j) = \frac{\sum_{i=0}^{n} R_i 1_{\pi(i)=a_j}}{n_j}$

- Greedy policy - pick action that currently gives best reward, $\pi(t) = \arg\max_a Q_t(a)$
- ϵ-greedy algorithm - with ϵ probability, pick another arm randomly.



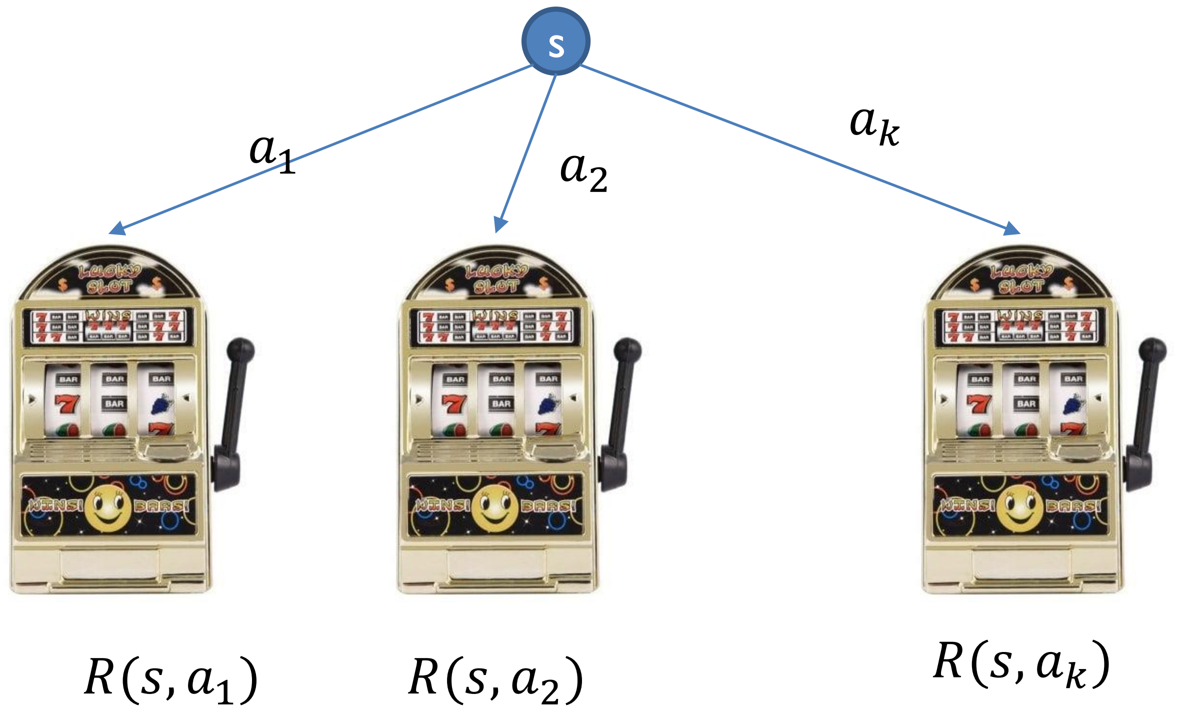
Multi-Armed bandit – Regret Minimization

- **Task:** find arm-pulling strategy such that the expected total reward at time n is close to the best possible.
  - Uniform Bandit – bad choice, wastes time with bad arms
  - Need to balance exploitation of good arms with exploration of poorly understood arms.



UCB Adaptive Bandit Algortihm

- **Task:** find arm-pulling strategy such that the expected total reward at time n is close to the best possible.
  - Uniform Bandit – bad choice, wastes time with bad arms
  - Need to balance exploitation of good arms with exploration of poorly understood arms.



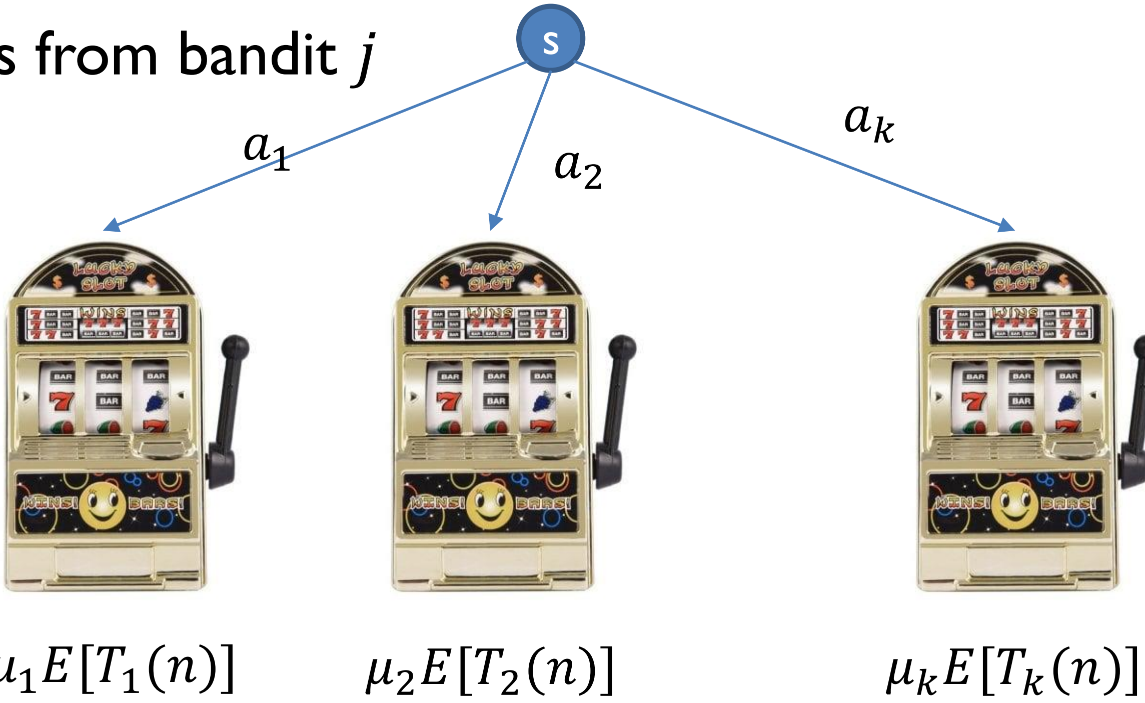
Regret

- Aiming at "reward as close as possible to the best reward" means we are minimizing **regret:**

$$R_n = \mu^* n - \sum_{j=1}^{k} \mu_j E[T_j(n)]$$

Where μ_j are the expected payoffs of arms, μ^* is the best expected payoff and $E[T_j(n)]$ is the expected number of pulls on arm j in total n pulls.

- $X_{j,1}, X_{j,2}$ ... = i.i.d r.v. of rewards from bandit j
- μ_j= expected value of X_j



Minimizing regret - UCB

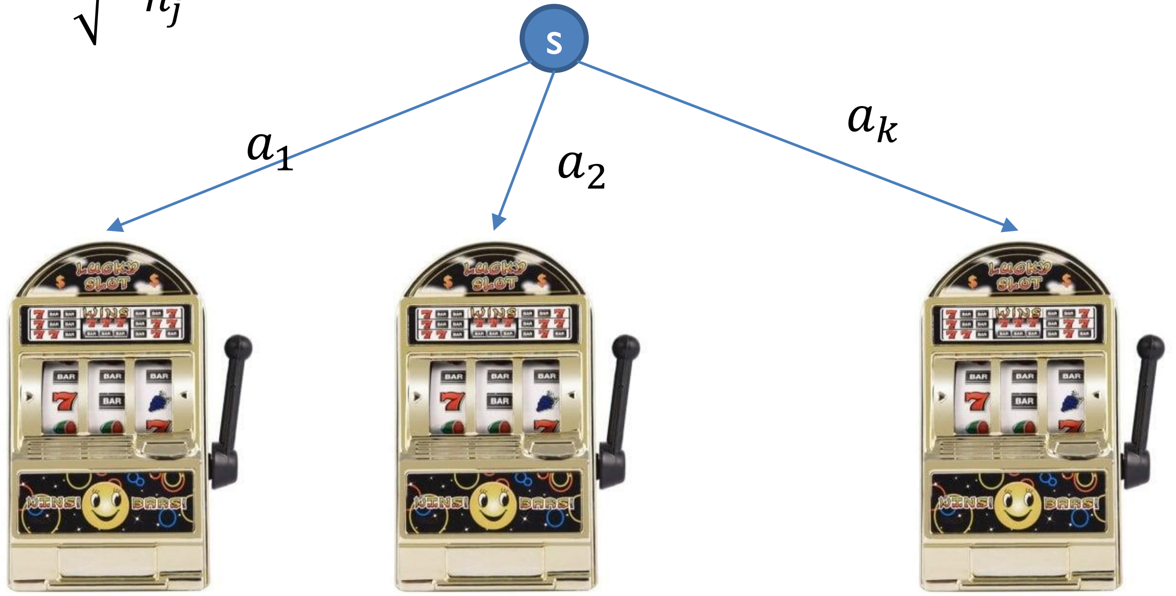
- Upper Confidence Bounds [Auer er. al., 2002]:

$$\boxed{UCB = \underbrace{\overline{X}_j}_{\text{Exploitation}} + \underbrace{\sqrt{\frac{2\ln n}{n_j}}}_{\text{Exploration}}}$$

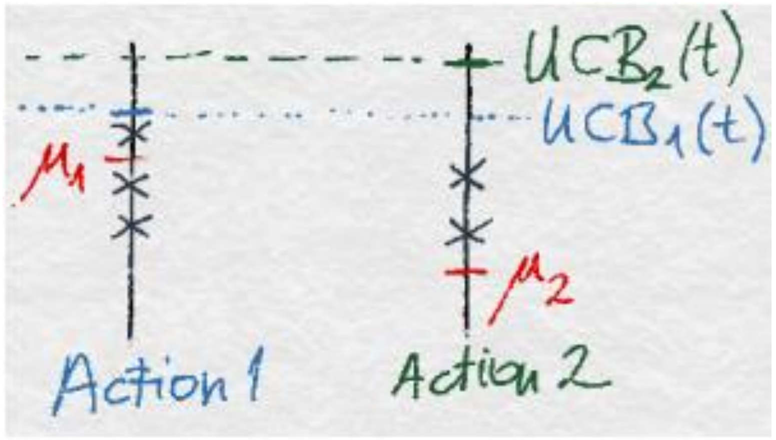
- When choosing arm, always select arm with highest UCB value
- $\overline{X}_j$ = mean of observed rewards, n = number of plays so far
- Using UCB, regret is upper bound by O(ln(n))

UCB - Example

$$\bar{X}_j + \sqrt{\frac{2\ln n}{n_j}}$$



- Play all arms once initially
- Then based on the formula

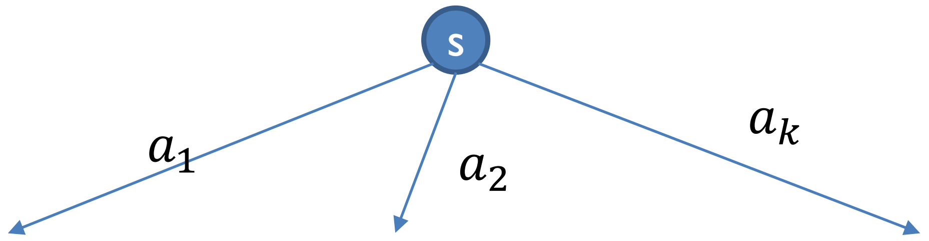


UCB - Example

$$\bar{X}_j + \sqrt{\frac{2 \ln n}{n_j}}$$

- $\sqrt{\frac{2 \ln n}{n_j}}$ is based of bound of the form $P\left(\bar{X}_j - E[X] \geq f(\sigma, n)\right) \leq \sigma$ (Remember PAC?)
- And σ is chosen to be time dependent (by n), goes to zero.

UCB



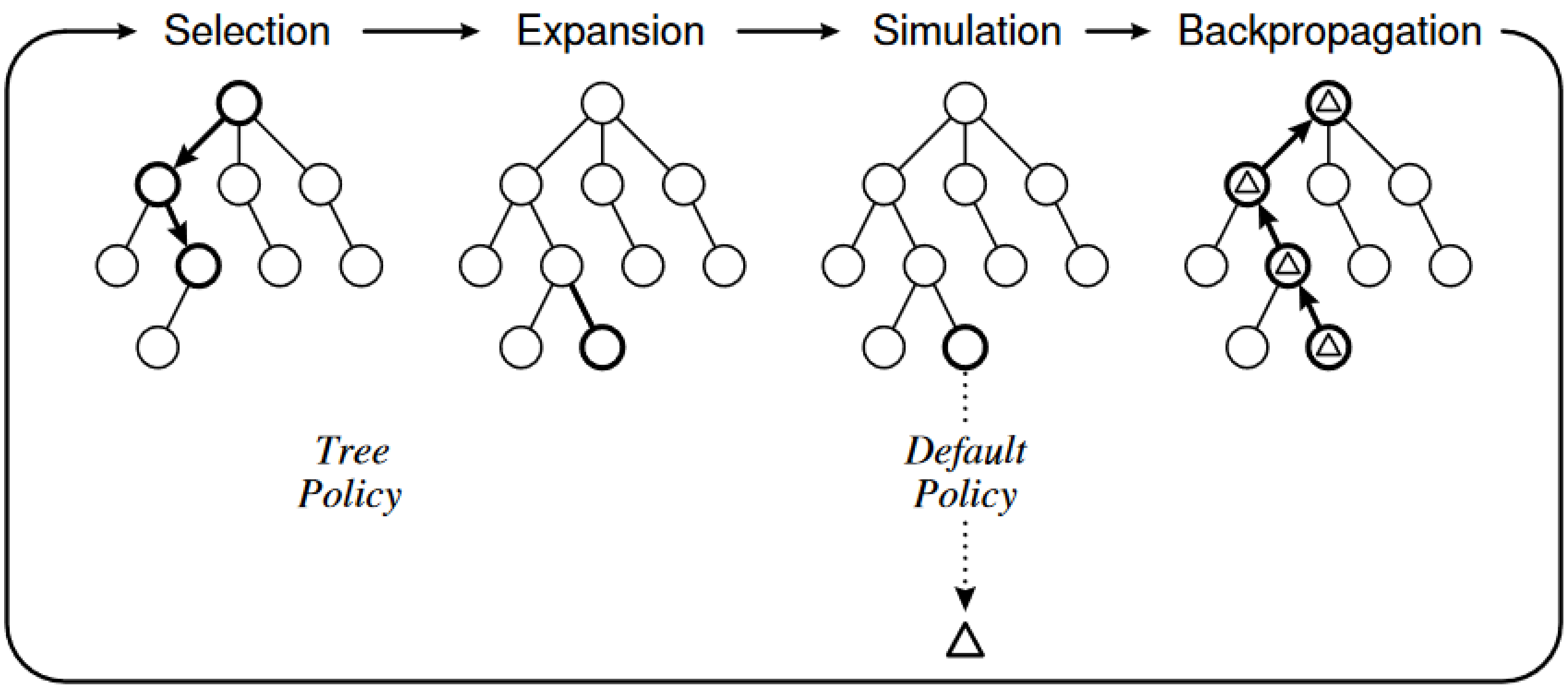
Excel example:

https://drive.google.com/open?id=1A9Kr-JDz_ZJlYOX3aFMrFaLUAPeAZV7Z

Google sheets:

https://docs.google.com/spreadsheets/d/17xxXMAGbXqjt6N1tah3VwKbusz5c44kGcAWQuhV93P0/edit?usp=sharing

UCB for Trees = UCT



•Tree node:

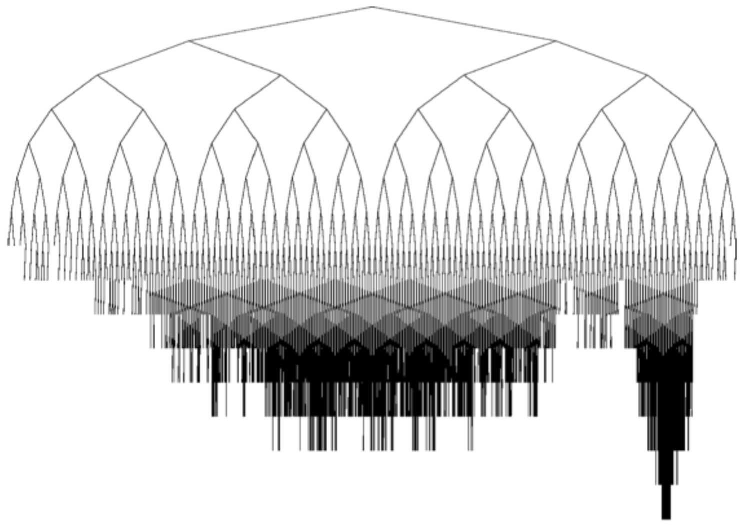
- Associated state,
- incoming action,
- number of visits,
- accumulated reward

•External slides by Michele Sebag:
https://drive.google.com/open?id=1ytp9l33_6WNe62qLAzV326iS4WmYeFpY

MCTS notes

- Aheuristic
  - Does not require any domain specific knowledge
  - Domain specific knowledge can provide significant speedups
- Anytime
  - Can return currently best action when stopped at any time
- Asymmetric
  - Tree is not explored fully
- MCTS = UCT? No consistency

  in the naming


[Arnaud et al., 2007]

Michele Sebag – MCTS slides

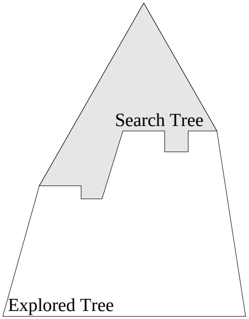
- External slides by Michele Sebag: https://drive.google.com/open?id=1ytp9l33_6WNe62qLAzV326iS4WmYeFpY

Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action
      Bandit phase
    - Add a node
      Grow a leaf of the search tree
    - Select next action bis
      Random phase, roll-out
    - Compute instant reward
      Evaluate
    - Update information in visited nodes
      Propagate
- Returned solution:
  - Path visited most often



Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action

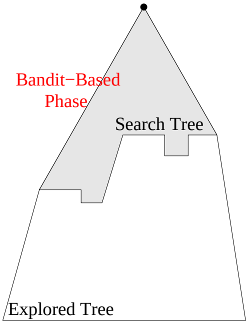
      Bandit phase
    - Add a node

      Grow a leaf of the search tree
    - Select next action bis

      Random phase, roll-out
    - Compute instant reward

      Evaluate
    - Update information in visited nodes

      Propagate
- Returned solution:
  - Path visited most often

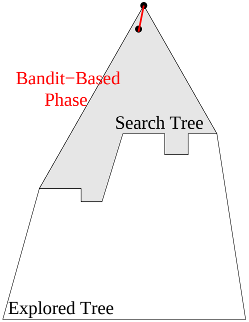


Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action
      Bandit phase
    - Add a node
      Grow a leaf of the search tree
    - Select next action bis
      Random phase, roll-out
    - Compute instant reward
      Evaluate
    - Update information in visited nodes
      Propagate
- Returned solution:
  - Path visited most often

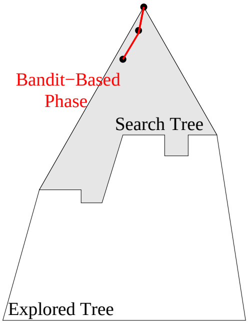


Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action
      Bandit phase
    - Add a node
      Grow a leaf of the search tree
    - Select next action bis
      Random phase, roll-out
    - Compute instant reward
      Evaluate
    - Update information in visited nodes
      Propagate
- Returned solution:
  - Path visited most often



Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action

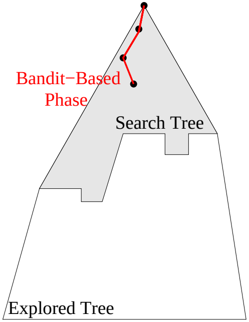
      Bandit phase
    - Add a node

      Grow a leaf of the search tree
    - Select next action bis

      Random phase, roll-out
    - Compute instant reward

      Evaluate
    - Update information in visited nodes

      Propagate
- Returned solution:
  - Path visited most often

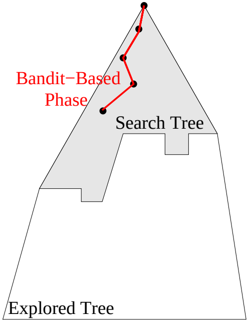


Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action
      Bandit phase
    - Add a node
      Grow a leaf of the search tree
    - Select next action bis
      Random phase, roll-out
    - Compute instant reward
      Evaluate
    - Update information in visited nodes
      Propagate
- Returned solution:
  - Path visited most often

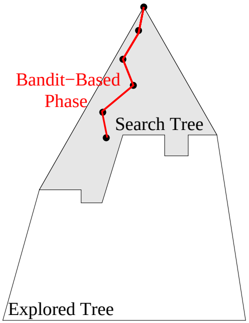


Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action
      Bandit phase
    - Add a node
      Grow a leaf of the search tree
    - Select next action bis
      Random phase, roll-out
    - Compute instant reward
      Evaluate
    - Update information in visited nodes
      Propagate
- Returned solution:
  - Path visited most often



Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action

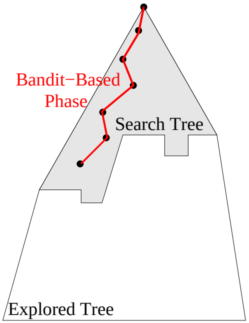
      Bandit phase
    - Add a node

      Grow a leaf of the search tree
    - Select next action bis

      Random phase, roll-out
    - Compute instant reward

      Evaluate
    - Update information in visited nodes

      Propagate
- Returned solution:
  - Path visited most often

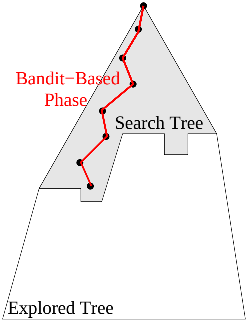


Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action
      Bandit phase
    - Add a node
      Grow a leaf of the search tree
    - Select next action bis
      Random phase, roll-out
    - Compute instant reward
      Evaluate
    - Update information in visited nodes
      Propagate
- Returned solution:
  - Path visited most often

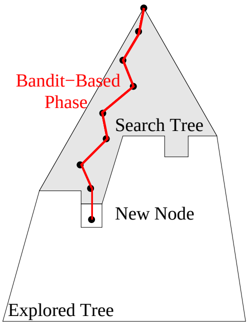


Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action
      Bandit phase
    - Add a node
      Grow a leaf of the search tree
    - Select next action bis
      Random phase, roll-out
    - Compute instant reward
      Evaluate
    - Update information in visited nodes
      Propagate
- Returned solution:
  - Path visited most often

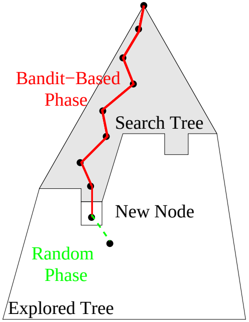


Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action
      Bandit phase
    - Add a node
      Grow a leaf of the search tree
    - Select next action bis
      Random phase, roll-out
    - Compute instant reward
      Evaluate
    - Update information in visited nodes
      Propagate
- Returned solution:
  - Path visited most often



Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action

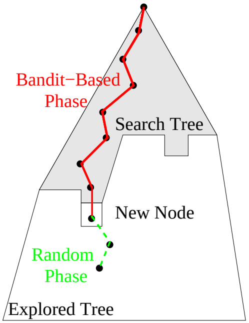
      Bandit phase
    - Add a node

      Grow a leaf of the search tree
    - Select next action bis

      Random phase, roll-out
    - Compute instant reward

      Evaluate
    - Update information in visited nodes

      Propagate
- Returned solution:
  - Path visited most often

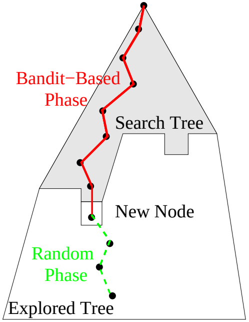


Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action
      Bandit phase
    - Add a node
      Grow a leaf of the search tree
    - Select next action bis
      Random phase, roll-out
    - Compute instant reward
      Evaluate
    - Update information in visited nodes
      Propagate
- Returned solution:
  - Path visited most often



Monte-Carlo Tree Search

Kocsis Szepesvári, 06

Gradually grow the search tree:

- Iterate Tree-Walk
  - Building Blocks
    - Select next action

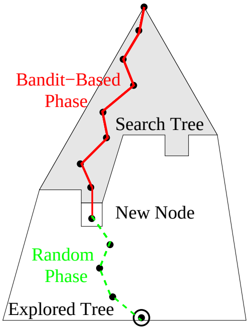
      Bandit phase
    - Add a node

      Grow a leaf of the search tree
    - Select next action bis

      Random phase, roll-out
    - Compute instant reward

      Evaluate
    - Update information in visited nodes

      Propagate
- Returned solution:
  - Path visited most often


Bandit−Based
Phase
Search Tree
New Node
Random
Phase
Explored Tree