

# MCTS Based on Simple Regret

David Tolpin, Solomon Eyal Shimony

Ben-Gurion University of the Negev  
Beer Sheva, Israel

February 22, 2012

# Hard to solve search problems

Search problems are often hard too solve in practice when:

- ▶ search space is extremely large;
- ▶ *and* good heuristics are unknown.

**Easier** to solve:

- ▶ Chess — search space size is manageable ( $10^{50}$ ).
- ▶ Timetabling — good heuristics.

**Hard** to solve:

- ▶ Compute Go ( $10^{180}$ ), Poker ( $10^{70}$ ).
- ▶ Canadian Traveller Problem.

# MCTS

**Monte Carlo Tree Search** helps in large search spaces.

- ▶ Starts with the root only.

**Adaptive** MCTS samples ‘good’ moves more frequently, but sometimes **explores** new directions.

# MCTS

**Monte Carlo Tree Search** helps in large search spaces.

- ▶ Starts with the root only.
- ▶ Repeats:
  1. **Selection:** select a branch to explore.

**Adaptive** MCTS samples 'good' moves more frequently, but sometimes **explores** new directions.

# MCTS

**Monte Carlo Tree Search** helps in large search spaces.

- ▶ Starts with the root only.
- ▶ Repeats:
  1. **Selection:** select a branch to explore.
  2. **Expansion:** adds children of the leaf to the stored tree.

**Adaptive** MCTS samples 'good' moves more frequently, but sometimes **explores** new directions.

# MCTS

**Monte Carlo Tree Search** helps in large search spaces.

- ▶ Starts with the root only.
- ▶ Repeats:
  1. **Selection:** select a branch to explore.
  2. **Expansion:** adds children of the leaf to the stored tree.
  3. **Simulation:** continues search (using a simple strategy) until a goal state is reached.

**Adaptive** MCTS samples 'good' moves more frequently, but sometimes **explores** new directions.

# MCTS

**Monte Carlo Tree Search** helps in large search spaces.

- ▶ Starts with the root only.
- ▶ Repeats:
  1. **Selection:** select a branch to explore.
  2. **Expansion:** adds children of the leaf to the stored tree.
  3. **Simulation:** continues search (using a simple strategy) until a goal state is reached.
  4. **Backpropagation:** values of each stored node are updated.

**Adaptive** MCTS samples 'good' moves more frequently, but sometimes **explores** new directions.

# Multi-armed Bandit Problem and UCB

Multi-armed Bandit Problem:

- ▶ We are given a set of  $K$  arms.
- ▶ Each arm can be pulled multiple times.
- ▶ The reward is drawn from an **unknown** (but normally *stationary* and *bounded*) distribution.
- ▶ The **total reward** must be maximized.

**UCB** is near-optimal for MAB — solves *exploration/exploitation* tradeoff.

- ▶ pulls an arm that maximizes **Upper Confidence Bound**:  
$$b_i = \bar{X}_i + \sqrt{\frac{c \log(n)}{n_i}}$$
- ▶ the cumulative regret is  $O(\log n)$ .



# UCT

UCT (**U**pper **C**onfidence Bounds applied to **T**rees) is based on UCB.

- ▶ Adaptive MCTS.
- ▶ Applies the UCB selection scheme at each step of the rollout.
- ▶ Demonstrated good performance in Computer Go (MoGo, CrazyStone, Fuego, Pachi, ...) as well as in other domains.

However, the first step of a rollout is different:

- ▶ The purpose of MCTS is to choose an action with the greatest utility.
- ▶ Therefore, the **simple regret** must be minimized.

Simple **R**egret followed by **C**umulative **R**egret.

- ▶ Maximizes **simple regret** at the **first step**.
- ▶ Continues with UCT from the **second step on**.

```

1: procedure ROLLOUT(node, depth=1)
2:   if ISLEAF(node, depth) then
3:     return 0
4:   else
5:     if depth=1 then action  $\leftarrow$  FIRSTACTION(node)
6:     else action  $\leftarrow$  NEXTACTION(node)
7:     next-node  $\leftarrow$  NEXTSTATE(node, action)
8:     reward  $\leftarrow$  REWARD(node, action, next-node)
9:           + ROLLOUT(next-node, depth+1)
10:    UPDATESTATS(node, action, reward)
  
```

# Sampling for Simple Regret

Sampling schemes for minimizing the simple regret:

1.  $\epsilon$ -greedy sampling.
2. a modified version of UCB (worse for cumulative, better for simple regret).
3. VOI-based sampling.

# Sampling for Simple Regret

Sampling schemes for minimizing the simple regret:

1.  $\epsilon$ -greedy sampling.
  2. a modified version of UCB (worse for cumulative, better for simple regret).
  3. VOI-based sampling.
- 1, 2 — heuristic selection criterion, theoretical upper bounds can be obtained.

# Sampling for Simple Regret

Sampling schemes for minimizing the simple regret:

1.  $\epsilon$ -greedy sampling.
  2. a modified version of UCB (worse for cumulative, better for simple regret).
  3. VOI-based sampling.
- ▶ 1, 2 — heuristic selection criterion, theoretical upper bounds can be obtained.
  - ▶ 3 — based on principles of *Rational Metareasoning*, but harder to analyze.

# Heuristic sampling schemes

$\varepsilon$ -greedy:

- ▶ Pulls the empirically best arm with probability  $\varepsilon$ .
- ▶ Any other arm with probability  $\frac{1-\varepsilon}{K-1}$ .
- ▶ Exhibits exponentially decreasing simple regret.
- ▶ Uniform sampling when  $\varepsilon = \frac{1}{K}$ , much better when  $\varepsilon = \frac{1}{2}$ .

# Heuristic sampling schemes

$\varepsilon$ -greedy:

- ▶ Pulls the empirically best arm with probability  $\varepsilon$ .
- ▶ Any other arm with probability  $\frac{1-\varepsilon}{K-1}$ .
- ▶ Exhibits exponentially decreasing simple regret.
- ▶ Uniform sampling when  $\varepsilon = \frac{1}{K}$ , much better when  $\varepsilon = \frac{1}{2}$ .

$UCB_{\sqrt{\cdot}}$  ( $\sqrt{\cdot}$  instead of  $\log$ ):

- ▶ Pulls arm  $i$  that maximizes  $b_i = \bar{X}_i + \sqrt{\frac{c\sqrt{n}}{n_i}}$ .
- ▶ Exhibits superpolynomially decreasing simple regret.

# VOI-aware sampling

- ▶ Chooses an action with the maximum VOI estimate.
- ▶ Estimates the VOI based on bounds on:
  - ▶ the probability that one or more rollouts will make another action appear better than the current best;
  - ▶ the gain that may be incurred if such a change occurs.

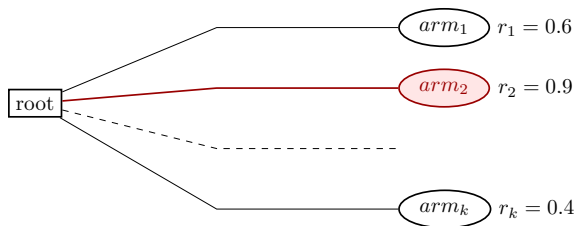
$$VOI_{\alpha} \approx \frac{\bar{X}_{\beta}}{n_{\alpha} + 1} \exp(-2(\bar{X}_{\alpha} - \bar{X}_{\beta})^2 n_{\alpha})$$

$$VOI_i \approx \frac{1 - \bar{X}_{\alpha}}{n_i + 1} \exp(-2(\bar{X}_{\alpha} - \bar{X}_i)^2 n_i), \quad i \neq \alpha$$

where  $\alpha = \arg \max_i \bar{X}_i, \quad \beta = \arg \max_{i, i \neq \alpha} \bar{X}_i$



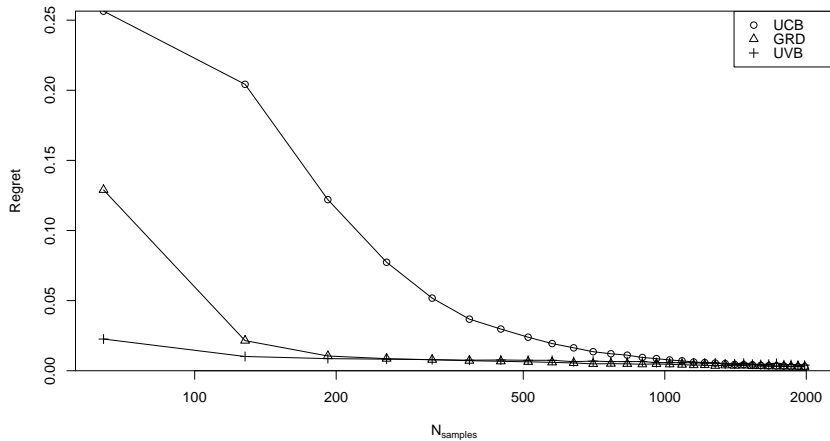
# Doing better than UCT on sets



When an arm is selected based on the **sample mean**:

- ▶ Regret of UCB decreases *polynomially* with  $n$ .
- ▶ Regret of  $\epsilon$ -greedy decreases *exponentially* with  $n$ .
- ▶ Regret of UVB:  $\max V_i$ ,  $V_{i_{best}} = \frac{1-1/k}{n_{i_{best}}}$ ,  $V_{i_{other}} = \frac{1/k}{n_{i_{other}}}$   
decreases exponentially with  $n$ , faster than  $\epsilon$ -greedy.

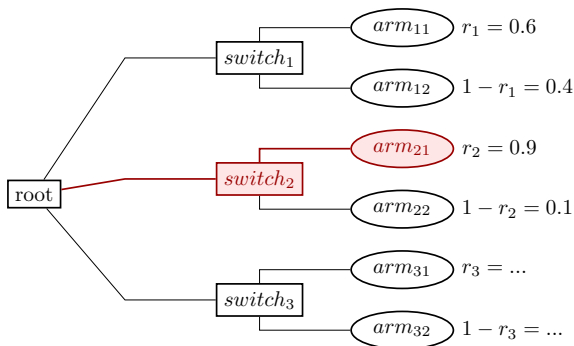
# UCB vs. $\epsilon$ -greedy vs UVB



64 Bernoulli arms, randomly generated

# Doing Better Than UCT on Trees

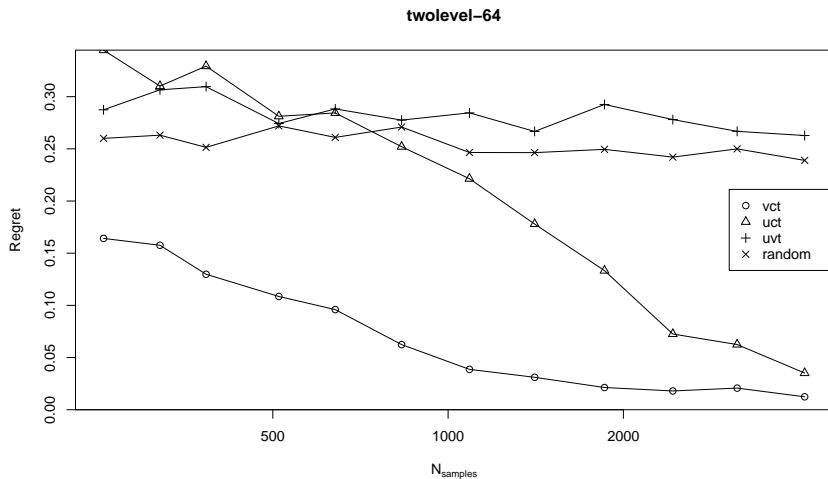
Uniform sampling is useless in this tree:



Rational sampling:

- ▶ first, choose an action that maximizes VOI (UVB);
- ▶ then, choose actions that maximize average reward (UCB).

# UVT vs. VCT (UVB+UCT) vs. UCT



64 Bernoulli arms, randomly generated

# Summary

- ▶ Improved MCTS scheme, SRCR, introduced.

# Summary

- ▶ Improved MCTS scheme, SRCR, introduced.
- ▶ SRCR performs better than unmodified UCT.

# Summary

- ▶ Improved MCTS scheme, SRCR, introduced.
- ▶ SRCR performs better than unmodified UCT.
- ▶ VOI-aware sampling for minimizing sampling regret proposed.

# Summary

- ▶ Improved MCTS scheme, SRCR, introduced.
- ▶ SRCR performs better than unmodified UCT.
- ▶ VOI-aware sampling for minimizing sampling regret proposed.
- ▶ Better sampling schemes can be developed based on principles of Rational Metareasoning.



# Summary

- ▶ Improved MCTS scheme, SRCR, introduced.
- ▶ SRCR performs better than unmodified UCT.
- ▶ VOI-aware sampling for minimizing sampling regret proposed.
- ▶ Better sampling schemes can be developed based on principles of Rational Metareasoning.
- ▶ UCT is not well understood. Better understanding will help in developing efficient MCTS schemes and adapting to different domains.

Thank you!