Beyond Classical Bandit Tools for Monte-Carlo Tree Search

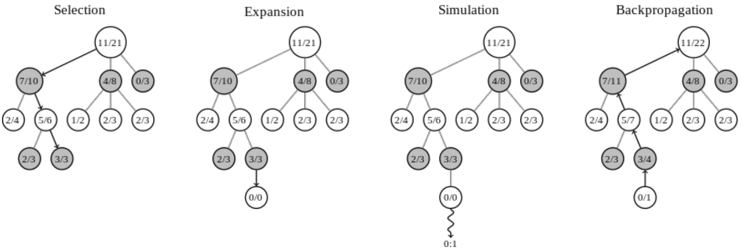
Emilie Kaufmann,

joint work with
Wouter M. Koolen (CWI) and Aurélien Garivier (ENS Lyon)

Centre de Recherche en Informatique, Signal et Automatique de Lille

AAAI Workshop on RL for Games,
Honolulu, January 28th, 2019

Playout-Based Monte-Carlo Tree Search



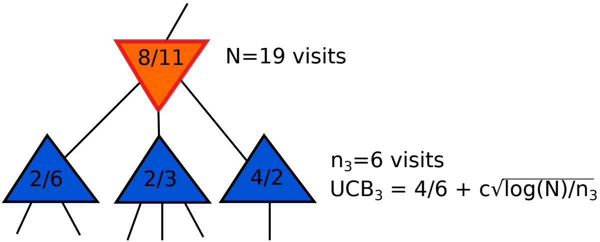
Goal: decide for the next move based on evaluation of possible trajectories in the game, ending with a random evaluation.

A famous bandit approach: [UCT, Koczis and Szepesvari 2006]

➜ use UCB in each node to decide the next children to explore

UCT for MCTS

Zoom on one (MAX) node and its children:



- UCT is *not* based on rigourous confidence intervals
- no sample complexity guarantees
- should we really *maximize rewards?*

Outline

μ_1 μ_2 μ_3 μ_4 μ_5

Goal: identify the arm with highest mean a^* (of mean μ^*)
(no loss when drawing "bad" arms)

The agent

- uses a sampling strategy : arm (A_t) is selected at round t
- stops at some (random) time τ
- upon stopping, recommends an arm $\hat{a}_\tau$

Formalization: an (ϵ, δ)-PAC algorithm:

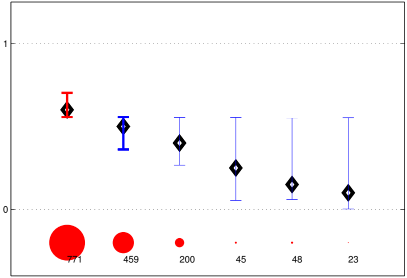
$$\mathbb{P}(\boldsymbol{\mu}_{\hat{\mathbf{a}}_\tau} \geq \boldsymbol{\mu}^* - \boldsymbol{\epsilon}) \geq \mathbf{1} - \boldsymbol{\delta}$$

with a small sample complexity $\boldsymbol{\tau}$. [Even Dar et al. 06]

The LUCB algorithm

An algorithm based on confidence intervals

$$\mathcal{I}_a(t) = [\mathrm{LCB}_a(t), \mathrm{UCB}_a(t)].$$



- At round t, draw

$$b_t = \underset{a}{\arg\max}\ \hat{\mu}_a(t)$$
$$c_t = \underset{a \neq b_t}{\arg\max}\ \mathrm{UCB}_a(t)$$

- Stop at round t if

$$\mathrm{LCB}_{b_t}(t) > \mathrm{UCB}_{c_t}(t) - \epsilon$$

Theorem [Kalyanakrishan et al. 2012]

For well-chosen confidence intervals, LUCB is (ϵ, δ)-PAC and

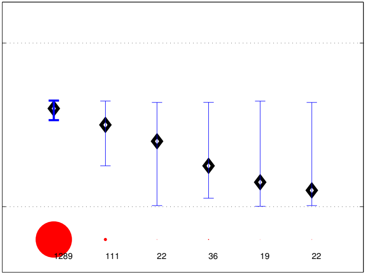
$$\mathbb{E}[\tau_\delta] = O\left(\left[\frac{1}{\Delta_2^2 \vee \epsilon^2} + \sum_{a=2}^{K} \frac{1}{\Delta_a^2 \vee \epsilon^2}\right] \ln\left(\frac{1}{\delta}\right)\right)$$

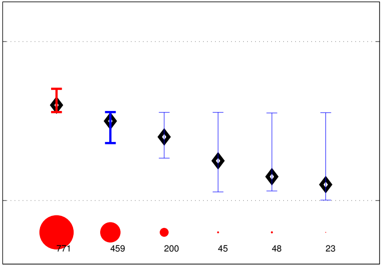
with $\Delta_a = \mu_1 - \mu_a$.

Regret minimization versus Best Arm Identification

Algorithms for regret minimization and BAI are very different!

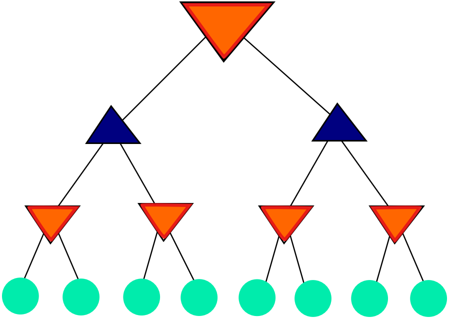
UCB versus LUCB





Outline

A simple model for MCTS



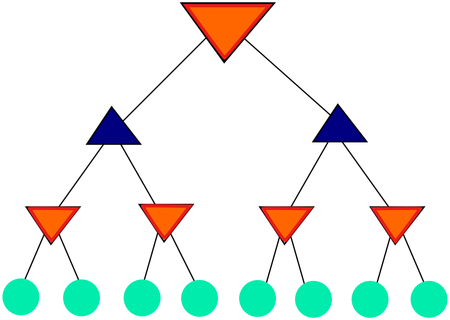
A fixed MAXMIN game tree $\mathcal{T}$, with leaves $\mathcal{L}$.

MAX node (= your move)

MIN node (= adversary move)

Leaf ℓ: stochastic oracle $\mathcal{O}_\ell$ that evaluates the position

A simple model for MCTS

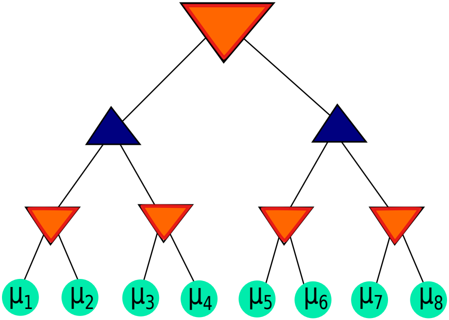


At round t a **MCTS algorithm**:

- picks a path down to a leaf L_t
- get an evaluation of this leaf $X_t \sim \mathcal{O}_{L_t}$

Assumption: i.i.d. sucessive evaluations, $\mathbb{E}_{X \sim \mathcal{O}_\ell}[X] = \mu_\ell$

A simple model for MCTS

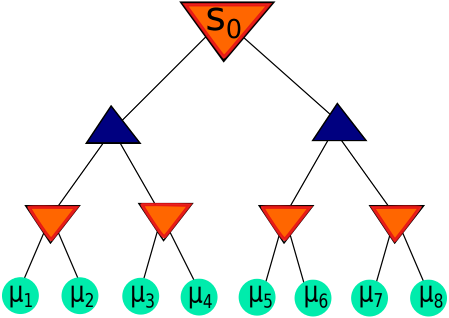


At round t a **MCTS algorithm**:

- picks a path down to a leaf L_t
- get an evaluation of this leaf $X_t \sim \mathcal{O}_{L_t}$

Assumption: i.i.d. sucessive evaluations, $\mathbb{E}_{X \sim \mathcal{O}_\ell}[X] = \mu_\ell$

Goal



A MCTS algorithm should find the best move at the root:

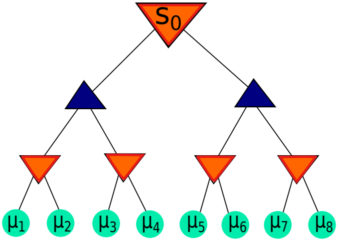
$$V_s = \begin{cases} \mu_s & \text{if s } \in \mathcal{L}, \\ \max_{c\in\mathcal{C}(s)} V_c & \text{if s is a MAX node,} \\ \min_{c\in\mathcal{C}(s)} V_c & \text{if s is a MIN node.} \end{cases}$$

$$s^* = \underset{s\in\mathcal{C}(s_0)}{\text{argmax}}\ V_s$$

A structured BAI problem

MCTS algorithm: $(L_t, \tau, \hat{s}_\tau)$, where

- L_t is the sampling rule
- τ is the stopping rule
- $\hat{s}_\tau \in \mathcal{C}(s_0)$ is the recommendation rule



Goal: an (ϵ, δ)-PAC MCTS algorithm:

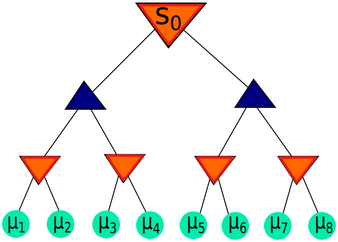
$$\mathbb{P}(\boldsymbol{V_{\hat{s}_\tau} \geq V_{s^*} - \epsilon}) \geq \boldsymbol{1 - \delta}$$

with a small sample complexity $\boldsymbol{\tau}$. [Teraoka et al. 14]

A structured BAI problem

MCTS algorithm: $(L_t, \tau, \hat{s}_\tau)$, where

- L_t is the sampling rule
- τ is the stopping rule
- $\hat{s}_\tau \in \mathcal{C}(s_0)$ is the recommendation rule



Idea: use LUCB on the depth-one nodes

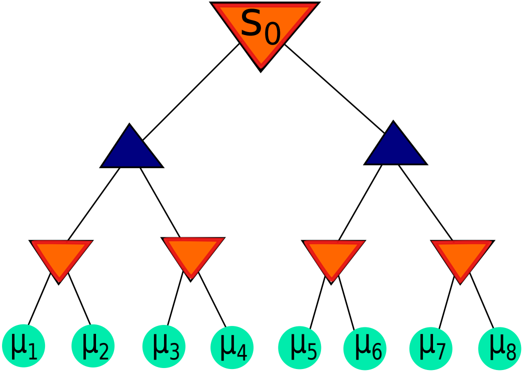
➜ requires confidence intervals on the values $(V_s)_{s \in \mathcal{C}_0}$

➜ requires to identify a leaf to sample starting from $s \in \mathcal{C}_0$

First tool: confidence intervals

Using the samples collected for the leaves, one can build, for $\ell \in \mathcal{L}$,

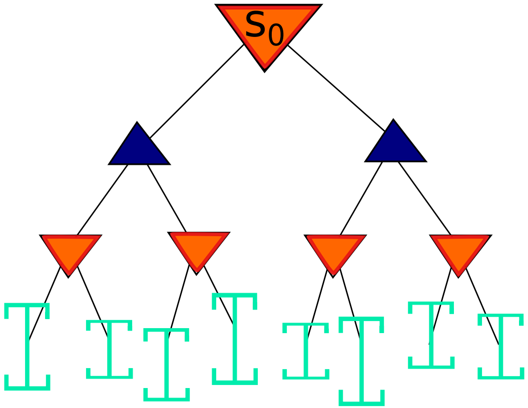
$$[\mathrm{LCB}_\ell(t), \mathrm{UCB}_\ell(t)] \quad \text{a confidence interval on } \mu_\ell$$



First tool: confidence intervals

Using the samples collected for the leaves, one can build, for $\ell \in \mathcal{L}$,

$[\mathrm{LCB}_\ell(t), \mathrm{UCB}_\ell(t)]$ a confidence interval on μ_ℓ

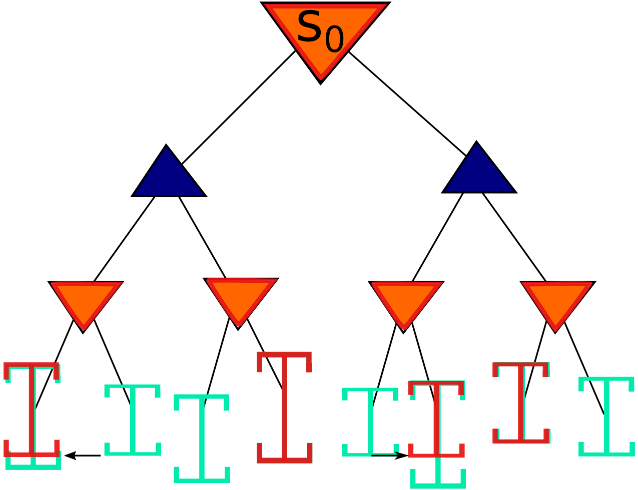


Idea: Propagate these confidence intervals up in the tree

First tool: confidence intervals

MAX node:

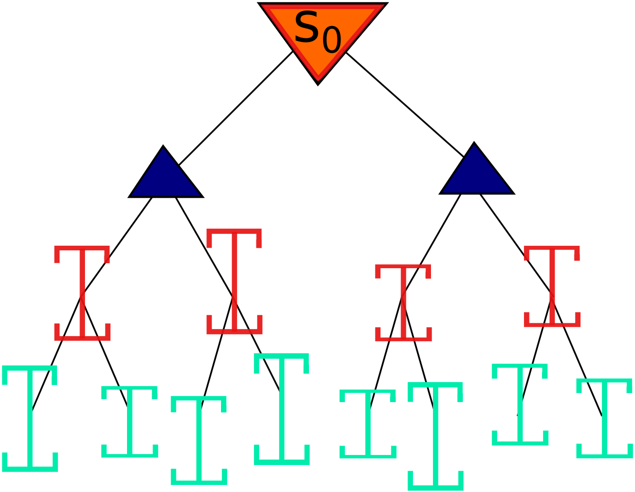
$$\mathrm{UCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{UCB}_c(t) \quad \mathrm{LCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{LCB}_c(t)$$



First tool: confidence intervals

MAX node:

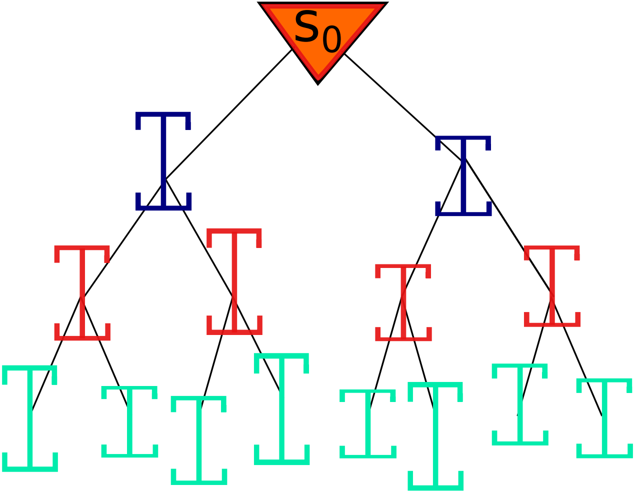
$$\mathrm{UCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{UCB}_c(t) \qquad \mathrm{LCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{LCB}_c(t)$$



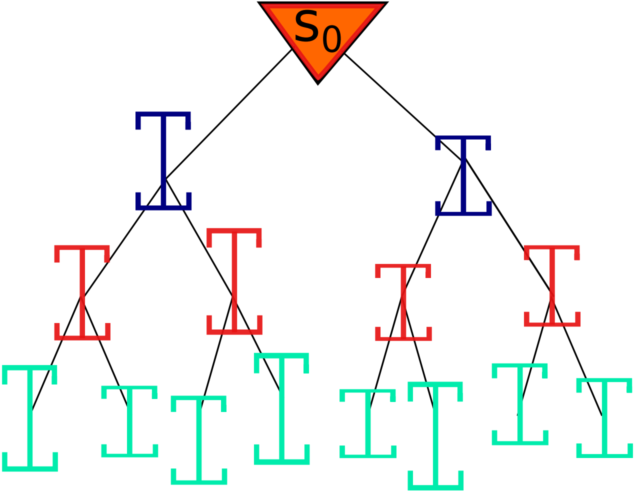
First tool: confidence intervals

MIN node:

$$\mathrm{UCB}_s(t) = \min_{c \in \mathcal{C}(s)} \mathrm{UCB}_c(t) \quad \mathrm{LCB}_s(t) = \min_{c \in \mathcal{C}(s)} \mathrm{LCB}_c(t)$$



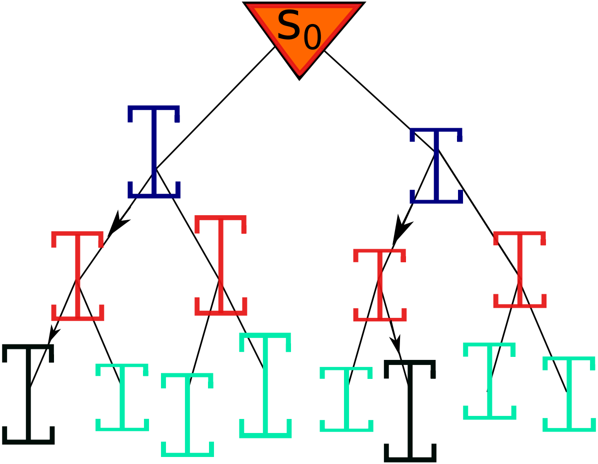
Property of this construction



$$\bigcap_{\ell \in \mathcal{L}} (\mu_\ell \in \mathcal{I}_\ell(t)) \quad \Rightarrow \quad \bigcap_{s \in \mathcal{T}} (V_s \in \mathcal{I}_s(t))$$

Second tool: representative leaves

$\ell_s(t)$: representative leaf of internal node $s \in \mathcal{T}$.



Idea: alternate optimistic/pessimistic moves starting from s

The BAI-MCTS architecture

- run a BAI algorithm on the depth-on nodes

$$\rightarrow \text{selects } R_t \in \mathcal{C}_0$$

- sample the representative leaf associated to that node:

$$L_t = \ell_{R_t}(t)$$

($\simeq$ starting from R_t, run UCT based on "true" CIs)

- update the confidence intervals
- stop when the BAI algorithm tell us to
- recommand the depth-one node chosen by the BAI algorithm

LUCB-MCTS

- Sampling rule: R_{t+1} is the least sampled among two promising depth-one nodes:

$$\underline{b}_t = \underset{s \in \mathcal{C}(s_0)}{\text{argmax}} \ \hat{V}_s(t) \quad \text{and} \quad \underline{c}_t = \underset{s \in \mathcal{C}(s_0) \setminus \{\underline{b}_t\}}{\text{argmax}} \ \mathrm{UCB}_s(t),$$

where $\hat{V}_s(t) = \hat{\mu}_{\ell_s(t)}(t)$. $\quad L_{t+1} = \ell_{R_{t+1}}(t)$.

- Stopping rule:

$$\tau = \inf \left\{ t \in \mathbb{N} : \mathrm{LCB}_{\underline{b}_t}(t) > \mathrm{UCB}_{\underline{c}_t}(t) - \epsilon \right\}$$

- Recommendation rule: $\hat{s}_\tau = \underline{b}_\tau$

Variant: UGapE-MCTS, based on [Gabillon et al. 12]

Theoretical guarantees

Given some exploration function $\beta(s,t)$, we choose confidence intervals of the form

$$\begin{aligned} \mathrm{LCB}_\ell(t) &= \hat{\mu}_\ell(t) - \sqrt{\frac{\beta(N_\ell(t),\delta)}{2N_\ell(t)}} \\ \mathrm{UCB}_\ell(t) &= \hat{\mu}_\ell(t) + \sqrt{\frac{\beta(N_\ell(t),\delta)}{2N_\ell(t)}}. \end{aligned}$$

Theorem [KK NIPS 17]

Choosing

$$\beta(s,\delta) \simeq \ln\left(\frac{|\mathcal{L}|\ln(s)}{\delta}\right),$$

LUCB-MCTS and UGapE-MCTS are (ϵ,δ)-PAC and

$$\mathbb{P}\left(\tau = O\left(H_\epsilon^*(\boldsymbol{\mu})\ln\left(\frac{1}{\delta}\right)\right)\right) \geq 1-\delta$$

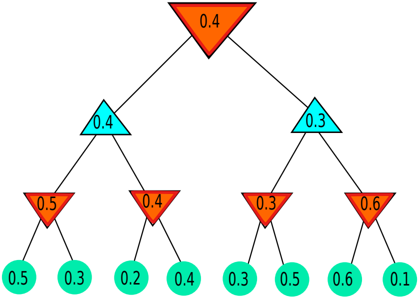
for UGapE-MCTS.

The complexity term

$$H^*_\epsilon(\mu) := \sum_{\ell \in \mathcal{L}} \frac{1}{\Delta_\ell^2 \vee \Delta_*^2 \vee \epsilon^2}$$

where

$$\begin{aligned} \Delta_* &:= V(s^*) - V(s_2^*) \\ \Delta_\ell &:= \max_{s \in \texttt{Ancestors}(\ell) \setminus \{s_0\}} \left| V_{\texttt{Parent}(s)} - V_s \right| \end{aligned}$$



Some room for improvements

We used the most naive way to build upper and lower confidence bounds on the minimum/maximum of several means

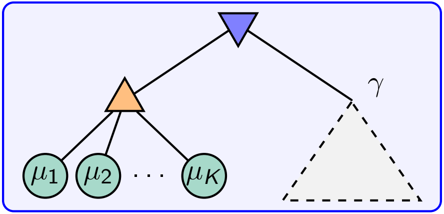
➜ use improved confidence intervals ?

One expects (and lower bounds reveal) a sparsity pattern, i.e. some leaves should be visited less than $\ln(1/\delta)$ times.

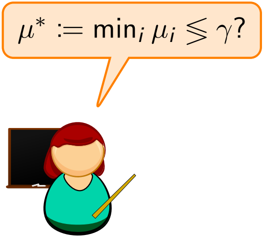
➜ can we derive optimal algorithms ?

Outline

Comparing a Node Value to a Threshold



Fix threshold γ.



For $t = 1, \ldots, \tau$

- pick a leaf A_t
- observe $X_t \sim \mu_{A_t}$

After stopping, recommend $\hat{m} \in \{<, >\}$

Goal: controlled error $\mathbb{P}_{\boldsymbol{\mu}} \{\text{error}\} < \delta$
and small sample complexity $\mathbb{E}_{\boldsymbol{\mu}}[\tau]$

Lower Bound and Oracle Allocation

Generic lower bound [Garivier et al. 16] shows *sample complexity* for *any* δ-correct algorithm is at least

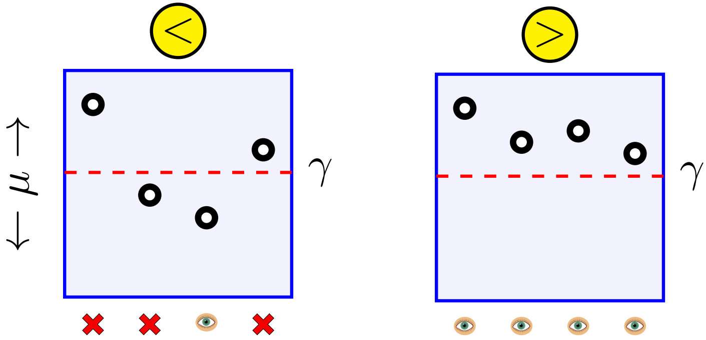
$$\mathbb{E}_{\boldsymbol{\mu}}[\tau] \;\geq\; T^*(\boldsymbol{\mu}) \ln\left(\frac{1}{\delta}\right).$$

For our problem the *characteristic time* and *oracle weights* are

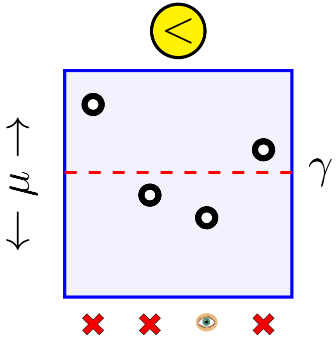
$$T^*(\boldsymbol{\mu}) = \begin{cases} \dfrac{1}{d(\mu^*,\gamma)} & \mu^* < \gamma, \\ \displaystyle\sum_a \dfrac{1}{d(\mu_a,\gamma)} & \mu^* > \gamma, \end{cases} \qquad w_a^*(\boldsymbol{\mu}) = \begin{cases} \mathbf{1}_{(a=a^*)} & \mu^* < \gamma, \\ \dfrac{\frac{1}{d(\mu_a,\gamma)}}{\sum_j \frac{1}{d(\mu_j,\gamma)}} & \mu^* > \gamma. \end{cases}$$

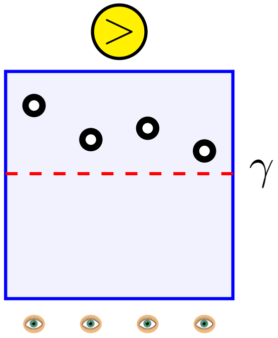
$w_a^*(\boldsymbol{\mu})$: fraction of selections of the leaf a under a strategy that would match the lower bound

Dichotomous Oracle Behaviour! Sampling Rule?



Dichotomous Oracle Behaviour! Sampling Rule?

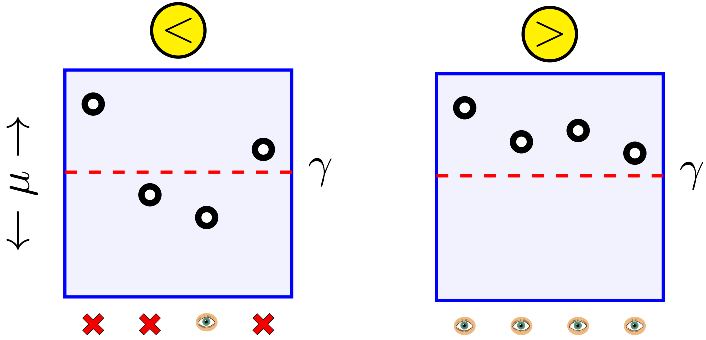




Two different ideas to get those sampling profiles:

- **Thompson Sampling** (Π_{t-1} is posterior after $t-1$ rounds)
  Sample $\theta \sim \Pi_{t-1}$, then play $A_t = \arg\min_a \ \theta_a$.
- **a Lower Confidence Bound algorithm**
  Play $A_t = \arg\min_a \ \mathrm{LCB}_a(t)$

A Solution: Murphy Sampling!



A more flexible idea:

- **Murphy Sampling** **condition on low minimum mean**
  Sample $\theta \sim \Pi_{t-1}\left(\cdot \middle| \min_a \theta_a < \gamma\right)$, then play $A_t = \arg\min_a \ \theta_a$.

$\rightarrow$ converges to the optimal allocation in both cases!

Properties of Murphy Sampling [KKG, NeurIPS'18]

Theorem

Asymptotic optimality: $N_a(t)/t \to w_a^*(\boldsymbol{\mu})$ *for all* $\boldsymbol{\mu}$

Sampling rule	$<$	$>$
Thompson Sampling	✓	✗
Lower Confidence Bounds	✗	✓
Murphy Sampling	✓	✓

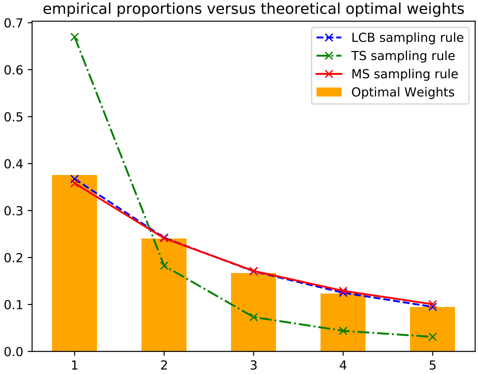
Lemma

Any anytime sampling strategy $(A_t)_t$ *ensuring* $\frac{\boldsymbol{N}_t}{t} \to \boldsymbol{w}^*(\boldsymbol{\mu})$ *and good stopping rule* τ_δ *guarantee* $\limsup_{\delta \to 0} \frac{\tau_\delta}{\ln \frac{1}{\delta}} \leq T^*(\boldsymbol{\mu})$.

$\to$ Murphy Sampling combined with a good stopping rule asymptotically attains the optimal sample complexity.

Numerical Results: proportions on $>$

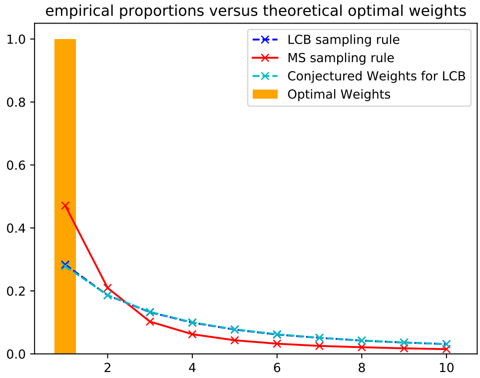
$$\boldsymbol{\mu} = \text{linspace}(1/2, 1, 5) \in \mathcal{H}_{>}$$



Sampling proportions vs oracle, $\delta = e^{-7}$.

Numerical Results: proportions on $<$

$$\boldsymbol{\mu} = \text{linspace}(-1, 1, 10) \in \mathcal{H}_<$$

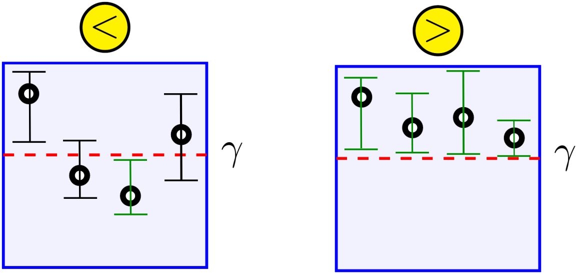


Sampling proportions vs oracle, $\delta = e^{-23}$.

What is a “good stopping rule”?

Example: a stopping rule based on individual confidence bounds:
$\tau^{\text{Box}} := \min(\tau_<; \tau_>)$ where

$$\begin{aligned}\tau_< &= \inf\{t \in \mathbb{N} : \exists a : \mathrm{UCB}_a(t) < \gamma\} \\ \tau_> &= \inf\{t \in \mathbb{N} : \forall a, \mathrm{LCB}_a(t) > \gamma\}\end{aligned}$$



$\tau = \tau_<$ $\qquad$ $\tau = \tau_>$

What is a "good stopping rule"?

Example: a stopping rule based on individual confidence bounds:
$\tau^{\text{Box}} := \min(\tau_<; \tau_>)$ where

$$\begin{aligned} \tau_< &= \inf\{t \in \mathbb{N} : \exists a : \mathrm{UCB}_a(t) < \gamma\} \\ \tau_> &= \inf\{t \in \mathbb{N} : \forall a, \mathrm{LCB}_a(t) > \gamma\} \end{aligned}$$

➜ enough to have the previous (asymptotic) results, but in practice we want to leverage the following:

Multiple low arms
identical or similar

$$\Rightarrow \begin{cases} \text{conclude } \mu^* < \gamma \textbf{ faster} \\ \textbf{tighter} \text{ confidence interval for } \mu^* \end{cases}$$

Improved Upper Confidence Bound on a Minimum

We identify a threshold function $T(x) = x + o(x)$ such that for every fixed subset $\mathcal{S} \subseteq [K]$, w.h.p. $\geq 1 - \delta$,

$$\forall t : \left[N_{\mathcal{S}}(t) d^{+}\left(\hat{\mu}_{\mathcal{S}}(t), \min_{a \in \mathcal{S}} \mu_a\right) - \ln \ln N_{\mathcal{S}}(t)\right]^{+} \leq T\left(\ln \frac{1}{\delta}\right).$$

where $N_{\mathcal{S}}$ and $\hat{\mu}_{\mathcal{S}}(t)$ **aggregate all the samples from arms in** $\mathcal{S}$.

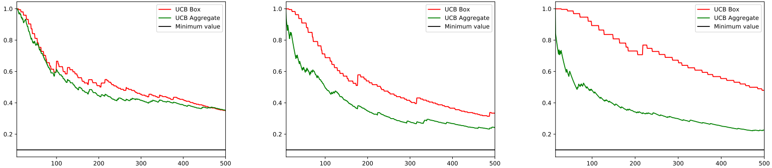
➜ yields a new upper confidence bound on μ^*

$$\mathrm{UCB}^{\pi}_{\min}(t) := \max\left\{q : \exists \mathcal{S} \subseteq [K] : \left[N_{\mathcal{S}} d^{+}(\hat{\mu}_{\mathcal{S}}, q) - \ln \ln N_{\mathcal{S}}\right] \leq T\left(\ln \frac{1}{\delta \pi(\mathcal{S})}\right)\right\},$$

and the corresponding stopping rule

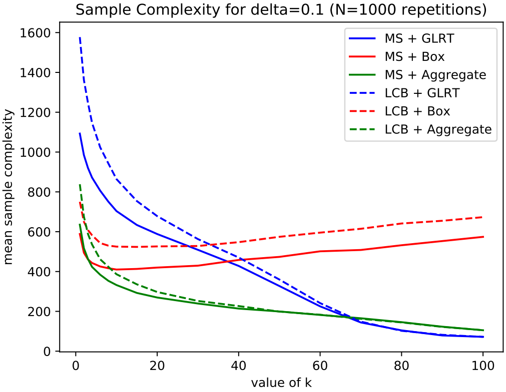
$$\tau_{<} = \inf\{t \in \mathbb{N} : \mathrm{UCB}^{\pi}_{\min}(t) \leq \gamma\}$$

Illustration of the new confidence regions



UCB for minimum: Agg *dominates* Box with 1, 3 and 10 low arms.

Sample Complexity Results



Agg *beats* Box and GLRT in adapting to the number k of low arms. Here $\mu_a \in \{-1, 0\}$ and $\gamma = 0$.

Future work

- use Murphy Sampling or our improved confidence intervals within an MCTS algorithm?
- handle growing trees
- [non MCTS related] propose efficient algorithms for more general *active testing* problems

References

Best Arm Identification

- Even-Dar et al., Action Elimination and Stopping Conditions for the Multi-Armed Bandit and Reinforcement Learning Problems. JMLR 2006.
- Kalyanakrishan et al., PAC subset selection in stochastic multi-armed bandits. ICML, 2012.
- K. and Garivier, Optimal Best Arm Identification with Fixed Confidence, COLT 2016
- Russo, Simple Bayesian Algorithms for Best Arm Identification, COLT 2016

BAI for games

- Teraoka et al., Efficient sampling method for Monte Carlo tree search problem. IEICE Transac. on Info. and Systems, 2014.
- Huang et al., Structured best arm identification with fixed confidence, ALT 2017.
- K. and Koolen, Monte-Carlo Tree Search by Best Arm Identification, NIPS 2017
- K., Koolen and Garivier, Sequential Test for the Lowest Mean: from Thompson to Murphy Sampling, NeurIPS 2018