Bandits (for) Games

Emilie Kaufmann,

joint work with Wouter M. Koolen (CWI)

Paris Symposium on Game Theory,
Paris, June 13th, 2018

The multi-armed bandit model

K arms = K probability distributions (ν_a has mean μ_a)

ν_1 ν_2 ν_3 ν_4 ν_5

At round t, an agent:

- chooses an arm A_t
- observes a sample $X_t \sim \nu_{A_t}$

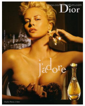
using a sequential sampling strategy (A_t):

$$A_{t+1} = F_t(A_1, X_1, \ldots, A_t, X_t).$$

Generic goal: learn the best arm, $a^* = \text{argmax}_a \, \mu_a$ of mean $\mu^* = \max_a \mu_a$

Bernoulli bandit model

K arms $=$ K **Bernoulli distributions**



$\mathcal{B}(\mu_1)$ $\mathcal{B}(\mu_2)$ $\mathcal{B}(\mu_3)$ $\mathcal{B}(\mu_4)$ $\mathcal{B}(\mu_5)$

At round t, an agent:

- chooses an arm A_t
- observes a sample $X_t \sim \mathcal{B}(\mu_{A_t})$: $\mathbb{P}(X_t = 1 | A_t) = \mu_{A_t}$

using a sequential sampling strategy (A_t):

$$A_{t+1} = F_t(A_1, X_1, \ldots, A_t, X_t).$$

Generic goal: learn the best arm, $a^* = \text{argmax}_a \ \mu_a$

Outline

1. First Bandit Game: Regret Minimization
2. Second Bandit Game: Best Arm Identification
3. Bandit Tools for Planning in Games

Outline

1. First Bandit Game: Regret Minimization
2. Second Bandit Game: Best Arm Identification
3. Bandit Tools for Planning in Games

Regret minimization in a bandit model

Samples = **rewards**, (A_t) is adjusted to

- maximize the (expected) sum of rewards,

$$\mathbb{E}\left[\sum_{t=1}^{T} X_t\right]$$

- or equivalently minimize the *regret*:

$$R_T = T\mu^* - \mathbb{E}\left[\sum_{t=1}^{T} X_t\right] = \sum_{a=1}^{K}(\mu^* - \mu_a)\mathbb{E}[N_a(T)]$$

$N_a(T)$: number of draws of arm a up to time T

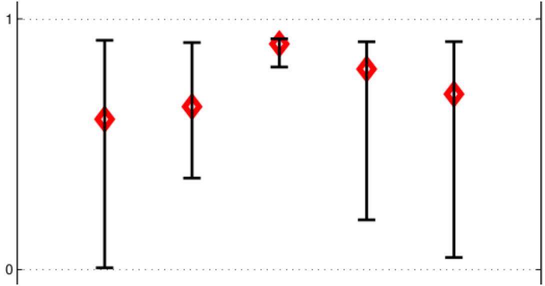
$\Rightarrow$ **Exploration/Exploitation tradeoff**
or... **Learning while Earning**

The UCB approach

- A UCB-type (or *optimistic*) algorithm chooses at round t

$$A_{t+1} = \underset{a=1...K}{\text{argmax}} \ \mathrm{UCB}_a(t).$$

where $\mathrm{UCB}_a(t)$ is an Upper Confidence Bound on μ_a.

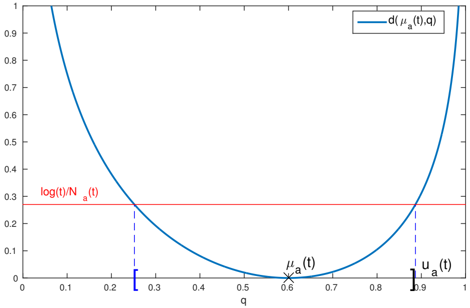


[Lai and Robbins 1985, Agrawal 1995, Auer et al. 02...]

The kl-UCB index

$$\mathrm{UCB}_a(t) := \max\left\{ q : d\left(\hat{\mu}_a(t), q\right) \leq \frac{\log(t)}{N_a(t)} \right\},$$

with $d(x, y) = \mathrm{KL}(\mathcal{B}(x), \mathcal{B}(y))$



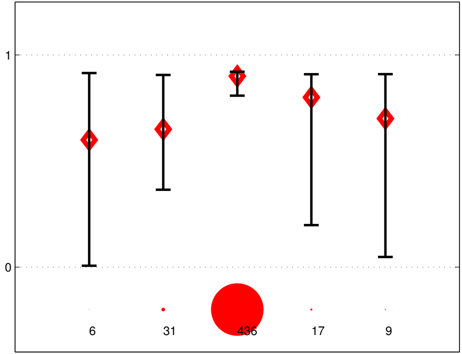
satisfies $\mathbb{P}(\mu_a \leq \mathrm{UCB}_a(t)) \gtrsim 1 - \frac{1}{t}$.

The kl-UCB algorithm

[Cappé et al. 13]: kl-UCB satisfies

$$\mathbb{E}_{\boldsymbol{\mu}}[N_a(T)] \leq \frac{1}{d(\mu_a, \mu^*)} \log T + O(\sqrt{\log(T)}).$$

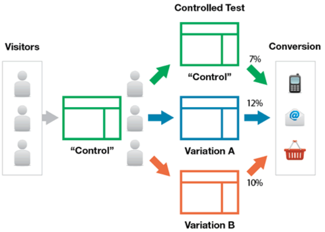
➜ matches the lower bound of [Lai and Robbins 1985]



Outline

A pure-exploration objective

Regret minimization:
maximize the number of conversions while learning which version of your webpage is the best



Alternative goal: quickly find out the best version for your webpage (no focus on conversions during the A/B testing phase)

Best arm identification

The agent has to identify the arm with highest mean a^* (no loss when drawing "bad" arms)

The agent

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

His goal:

Fixed-budget setting	Fixed-confidence setting
$\tau = T$ minimize $\mathbb{P}(\hat{a}_\tau \neq a^*)$	minimize $\mathbb{E}[\tau]$ $\mathbb{P}(\hat{a}_\tau \neq a^*) \leq \delta$

[Bubeck et al. 2010] [Even Dar et al. 2006]

Best arm identification

The agent has to identify the arm with highest mean a^*
(no loss when drawing "bad" arms)

The agent

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

His goal:

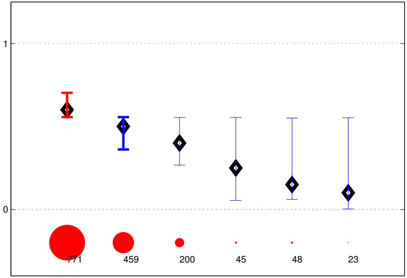
Fixed-budget setting	**Fixed-confidence setting**
$\tau = T$	**minimize** $\mathbb{E}[\boldsymbol{\tau}]$
minimize $\mathbb{P}(\hat{a}_\tau \neq a^*)$	$\mathbb{P}(\boldsymbol{\mu}_{\hat{\mathbf{a}}_{\boldsymbol{\tau}}} < \boldsymbol{\mu}^* - \boldsymbol{\epsilon}) \leq \boldsymbol{\delta}$

(ϵ, δ)-PAC algortihm

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]

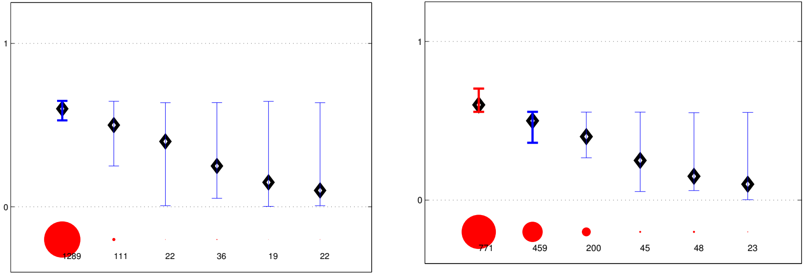
For well-chosen confidence intervals, LUCB is (ϵ, δ)-PAC and

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

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

Algorithms for regret minimization and BAI are very different!

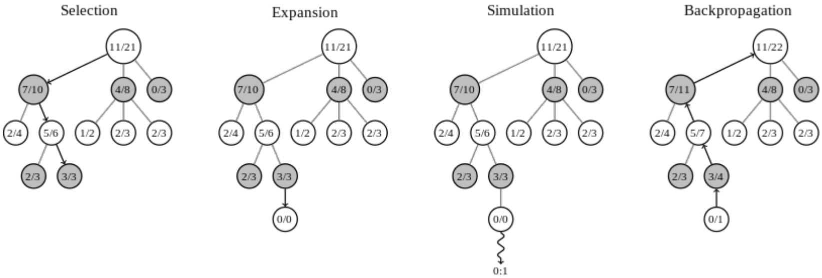
kl-UCB versus (kl)-LUCB



Next: how to use them for planning in games !

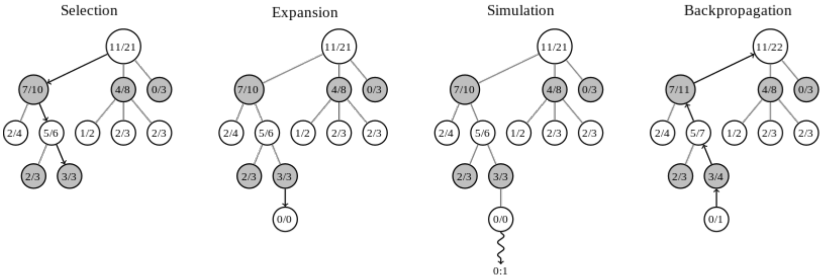
Outline

Monte-Carlo Tree Search for games



Goal: decide for the next move based on evaluation of possible trajectories in the game

Monte-Carlo Tree Search for games



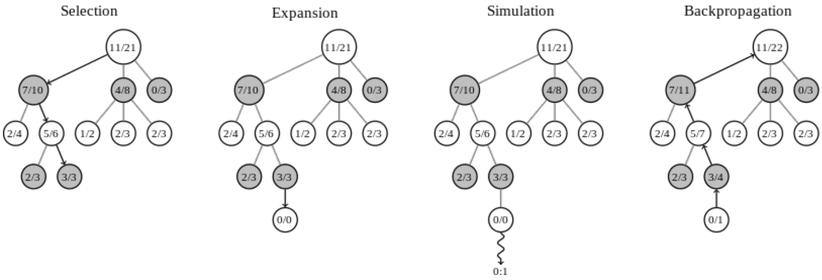
Goal: decide for the next move based on evaluation of possible trajectories in the game

Usual bandit approach: [UCT, Koczis and Szepesvari 2006]

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

➜ no sample complexity guarantees

Monte-Carlo Tree Search for games

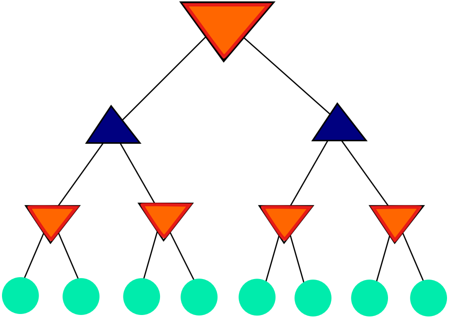


We introduce an idealized model:

- *fixed* maximin tree
- *i.i.d.* playouts starting from each leaf

and propose new algorithms with sample complexity guarantees

A simple model for MCTS



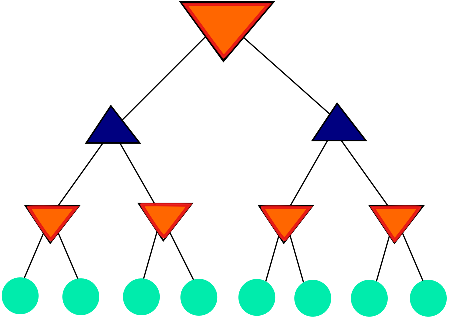
A fixed MAXMIN game tree $\mathcal{T}$, with leaves $\mathcal{L}$.

 MAX node (= your move)

 MIN node (= adversary move)

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

A simple model for MCTS



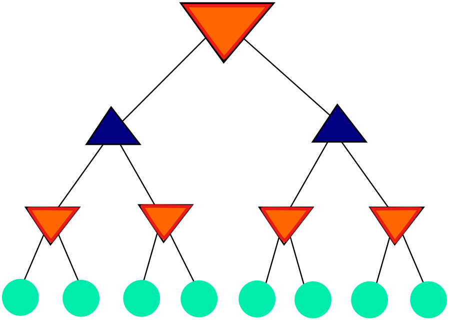
A fixed MAXMIN game tree $\mathcal{T}$, with leaves $\mathcal{L}$.

MAX node (= your move)

MIN node (= adversary move)

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

A simple model for MCTS

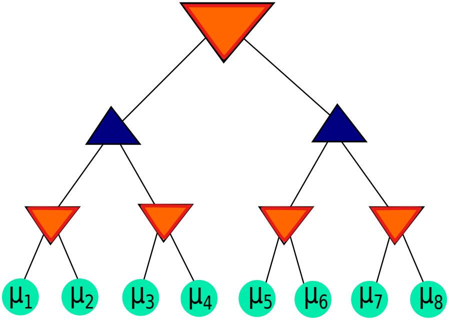


At round t a **MCTS algorithm**:

- 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

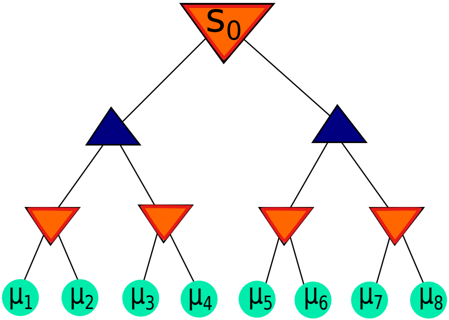


At round t a **MCTS algorithm**:

- 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

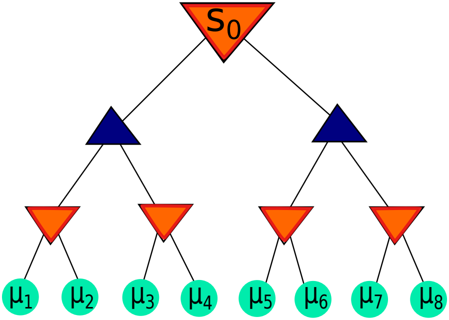


A MCTS algorithm should find the best move at the root:

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

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

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

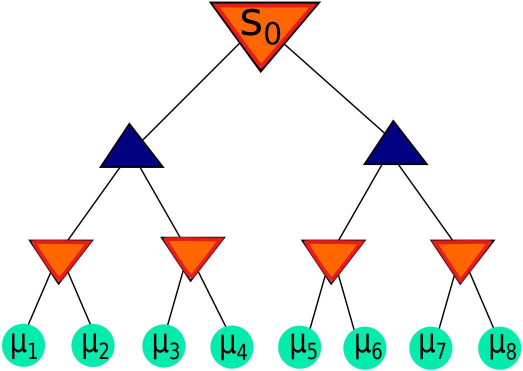
is $(\epsilon, \delta) -$ PAC if $\quad \mathbb{P}\left(V_{\hat{s}_\tau} \geq V_{s^*} - \epsilon\right) \geq 1 - \delta.$

Goal: (ϵ, δ)-PAC algorithm with a small sample complexity τ.

First tool: confidence intervals

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

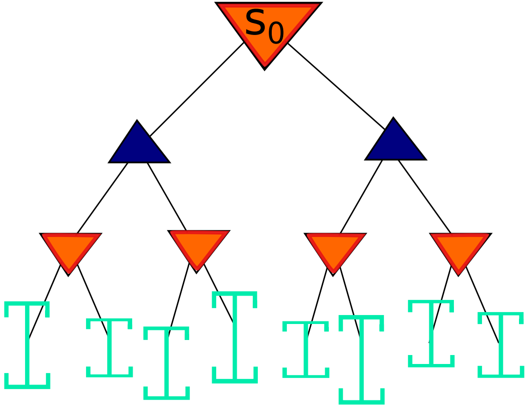
$[\mathrm{LCB}_\ell(t), \mathrm{UCB}_\ell(t)]$ a confidence interval on μ_ℓ



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

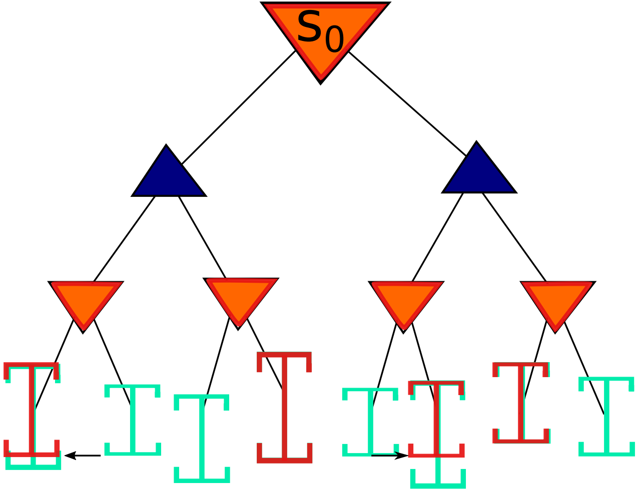


Idea: Propagate these confidence intervals up in the tree

First tool: confidence intervals

MAX node:

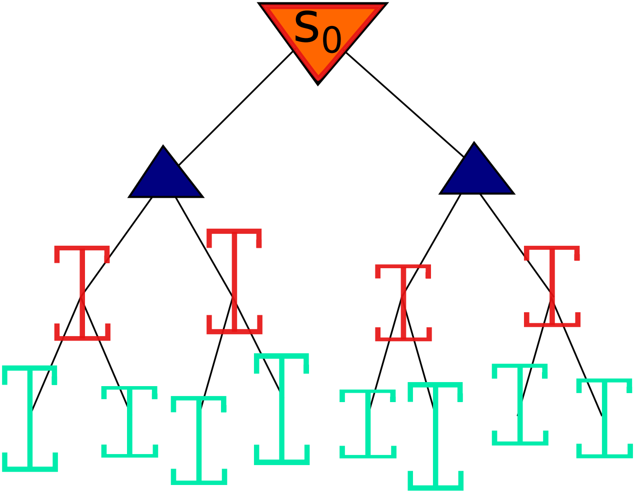
$$\mathrm{UCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{UCB}_c(t) \quad \mathrm{LCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{LCB}_c(t)$$



First tool: confidence intervals

MAX node:

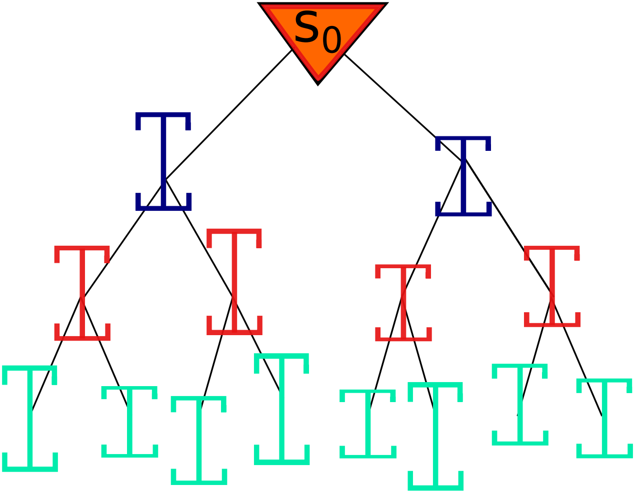
$$\mathrm{UCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{UCB}_c(t) \qquad \mathrm{LCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{LCB}_c(t)$$



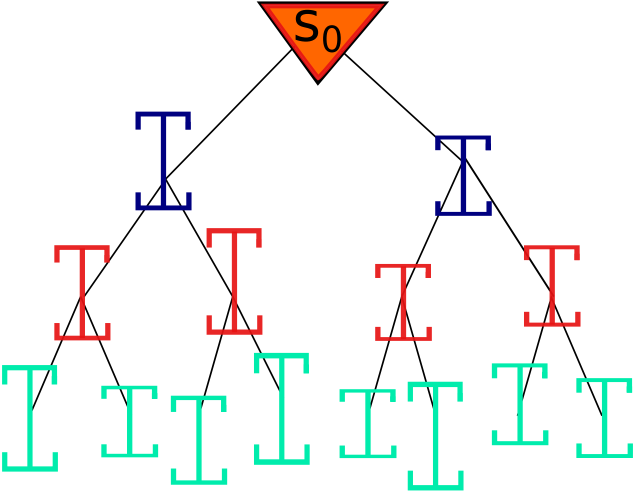
First tool: confidence intervals

MIN node:

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



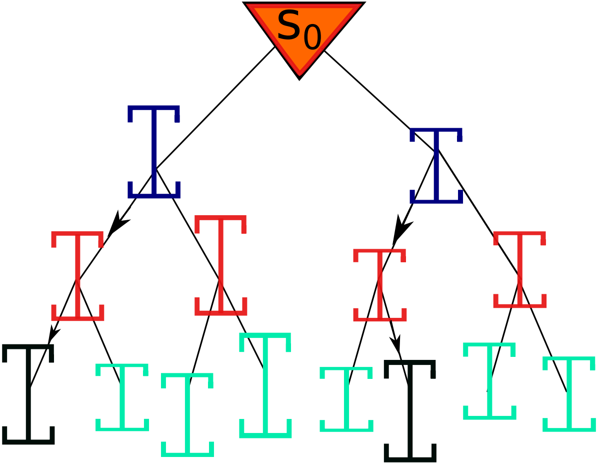
Property of this construction



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

Second tool: representative leaves

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



Idea: alternate optimistic/pessimistic moves starting from s

Generic BAI-MCTS algorithm

Input: a BAI algorithm
Initialization: $t = 0$.
while not $\mathbf{BAIStop}\left(\{s \in \mathcal{C}(s_0)\}\right)$ **do**

  $R_{t+1} = \mathbf{BAIStep}\left(\{s \in \mathcal{C}(s_0)\}\right)$
  Sample the representative leaf $L_{t+1} = \ell_{R_{t+1}}(t)$
  Update the information about the arms. $t = t + 1$.

end
Output: $\mathbf{BAIReco}\left(\{s \in \mathcal{C}(s_0)\}\right)$

Generic BAI-MCTS algorithm

Input: a BAI algorithm
Initialization: $t = 0$.
while not $\mathbf{BAIStop}\left(\{s \in \mathcal{C}(s_0)\}\right)$ **do**

$R_{t+1} = \mathbf{BAIStep}\left(\{s \in \mathcal{C}(s_0)\}\right)$

Sample the representative leaf $L_{t+1} = \ell_{R_{t+1}}(t)$

Update the information about the arms. $t = t + 1$.

end
Output: $\mathbf{BAIReco}\left(\{s \in \mathcal{C}(s_0)\}\right)$

... typically the confidence intervals

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 $\hat{V}_s(t) = \hat{\mu}_{\ell_s(t)}(t)$.

(empirical value of the representative leaf)

- 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

We choose confidence intervals of the form

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

where $\beta(s,\delta)$ is some exploration function.

Correctness

If $\delta \leq \max(0.1|\mathcal{L}|, 1)$, for the choice

$$\beta(s,\delta) = \log(|\mathcal{L}|/\delta) + 3\log\log(|\mathcal{L}|/\delta) + (3/2)\log(\log s + 1)$$

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

Theoretical guarantees

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

where

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

Sample complexity

With probability larger than $1-\delta$, the total number of leaves explorations performed by UGapE-MCTS is upper bounded as

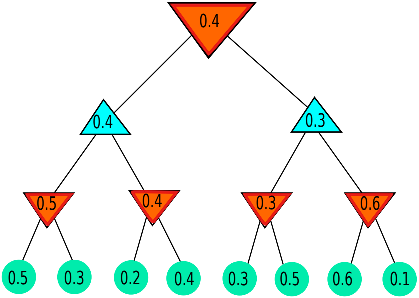
$$\tau = O\left(H^*_\epsilon(\mu)\log\left(\frac{1}{\delta}\right)\right).$$

Theoretical guarantees

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

where

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



Conclusion

Our contributions:

- a generic way to use a BAI algorithm for MCTS
- PAC and sample complexity guarantees for UGapE-MCTS and LUCB-MCTS...
- ... that also displays good empirical performance

Future work:

- identify the *optimal* sample complexity of the MCTS problem... (i.e. matching upper and lower bounds)
- ... and that of other structured Best Arm Identification problems [Huang et al., ALT 17]

Reference:

E. Kaufmann & W.M. Koolen,
Monte-Carlo Tree Search by Best Arm Identification
NIPS 2017