Monte-Carlo Tree Search by Best Arm Identification

Emilie Kaufmann and Wouter M. Koolen

Inria Lille
SequeL team

CWI
Machine Learning Group

Inria-CWI workshop
Amsterdam, September 20th, 2017

Part of...

... a new Associate Team proposal

[6]PAC

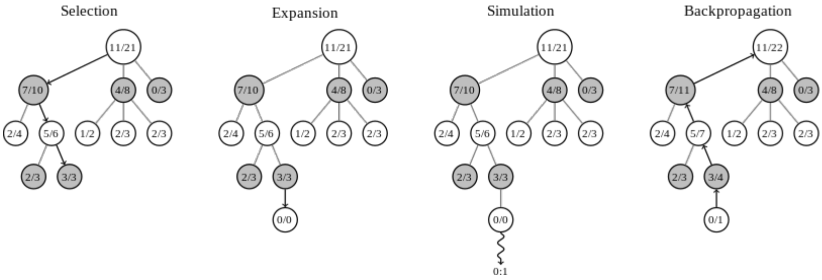
involving

- Peter Grünwald (CWI, Machine Learning Group)
- Wouter M. Koolen (CWI, Machine Learning Group)
- Benjamin Guedj (Inria Lille, MODAL project-team)
- Emilie Kaufmann (Inria Lille, SequeL project-team)

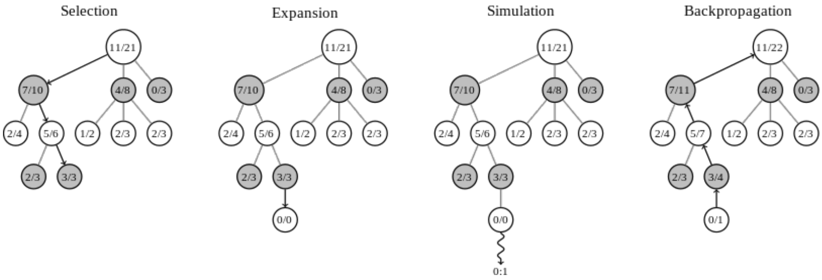
Broader goal: Probably Approximately Correct - Learning

[6] Safe, Efficient, Sequential, Active, Structured, Ideal

Monte-Carlo Tree Search for games



Monte-Carlo Tree Search for games



We introduce an idealized model:

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

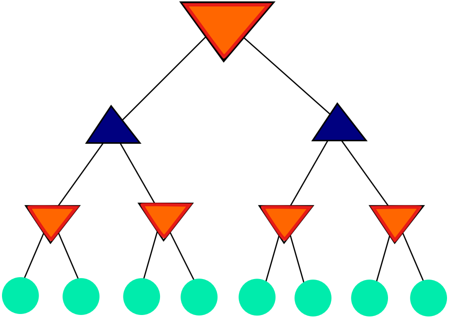
and propose new algorithms with sample complexity guarantees

Outline

1. Problem formulation
2. The BAI-MCTS architecture
3. UGapE-MCTS and LUCB-MCTS
4. Towards optimal algorithms

Outline

A simple model for MCTS



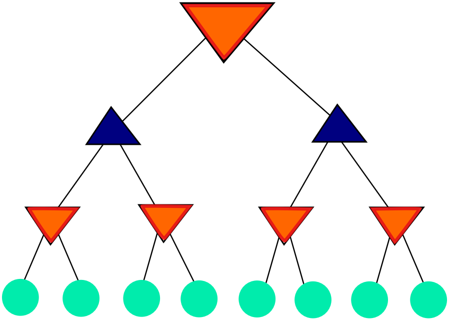
A fixed MAXMIN game tree $\mathcal{T}$, with leaves $\mathcal{L}$.

 MAX node (= your move)

 MIN node (= adversary move)

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

A simple model for MCTS

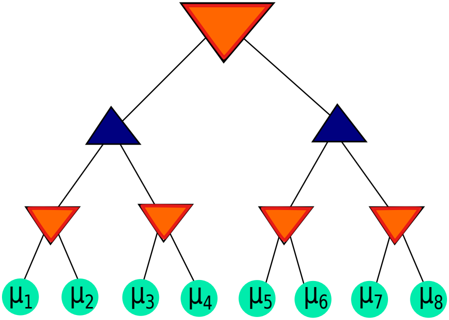


At round t a **MCTS algorithm**:

- 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, $\mathbb{E}_{X \sim \mathcal{O}_\ell}[X] = \mu_\ell$

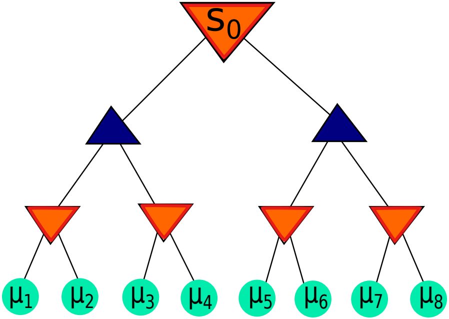
A simple model for MCTS



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, $\mathbb{E}_{X\sim\mathcal{O}_\ell}[X] = \mu_\ell$

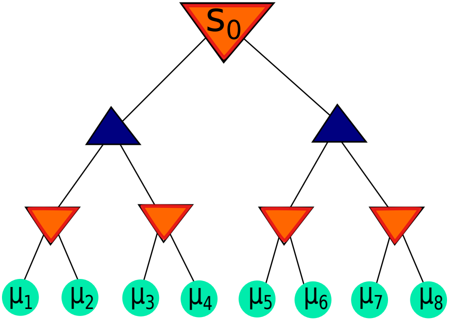


A MCTS algorithm should find the best move at the root:

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

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

A PAC learning framework



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.

A simpler problem: best arm identification

Reminiscent of a bandit model:

μ_1 μ_2 μ_3 μ_4 μ_5 μ_6 μ_7 μ_8

A Best Arm Identification algorithm: $(A_t, \tau, \hat{s}_\tau)$, where

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

is (ϵ, δ)-PAC if

$$\mathbb{P}\left(\mu_{\hat{s}_\tau} \geq \mu^* - \epsilon\right) \geq 1 - \delta.$$

A simpler problem: best arm identification

Reminiscent of a bandit model:

A Best Arm Identification algorithm: $(A_t, \tau, \hat{s}_\tau)$, where

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

is (ϵ, δ)-PAC if

$$\mathbb{P}\left(\mu_{\hat{s}_\tau} \geq \mu^* - \epsilon\right) \geq 1 - \delta.$$

The BAI problem:
How to adaptivly sample the arms so as to identify as quickly as possible the arm with highest mean ?

MCTS: a structured BAI problem

Reminiscent of a bandit model:

μ_1 μ_2 μ_3 μ_4 μ_5 μ_6 μ_7 μ_8

A Best Arm Identification 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 (ϵ, δ)-PAC if

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

The MCTS problem:

How to adaptivly sample the leaves of a maxmin tree so as to identify as quickly as possible the best action at the root ?

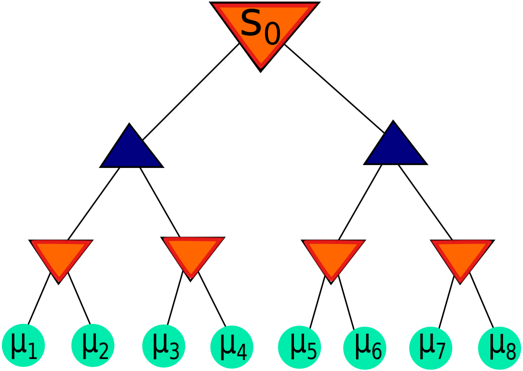
Outline

1. Problem formulation
2. The BAI-MCTS architecture
3. UGapE-MCTS and LUCB-MCTS
4. Towards optimal algorithms

A key building block: confidence intervals

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

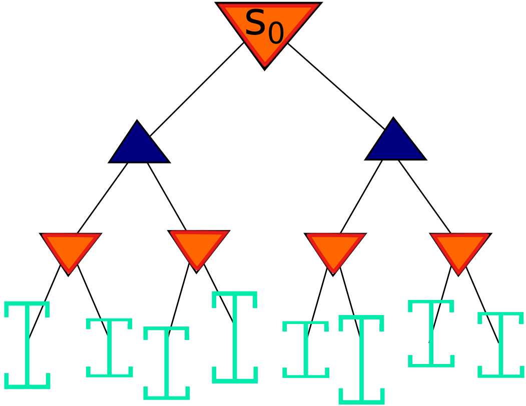
$$[\mathrm{LCB}_\ell(t), \mathrm{UCB}_\ell(t)] \quad \text{a confidence interval on } \mu_\ell$$



A key building block: 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

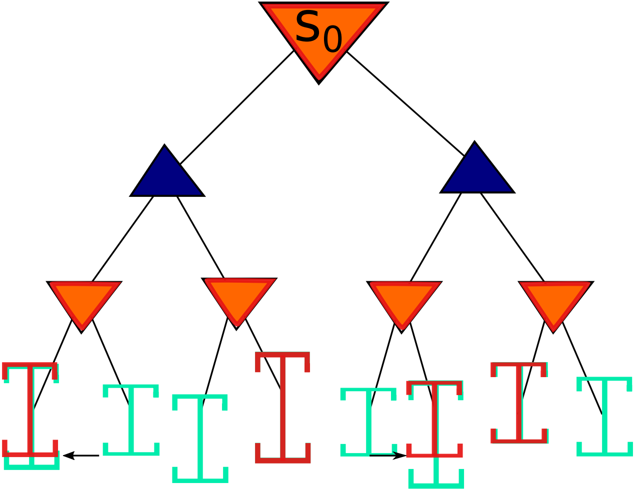


Idea: Propagate these confidence intervals up in the tree

A key building block: confidence intervals

MAX node:

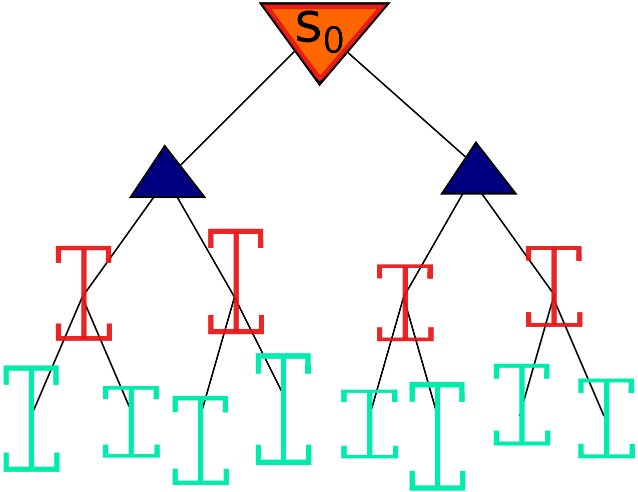
$$\mathrm{UCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{UCB}_c(t) \quad \mathrm{LCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{LCB}_c(t)$$



A key building block: confidence intervals

MAX node:

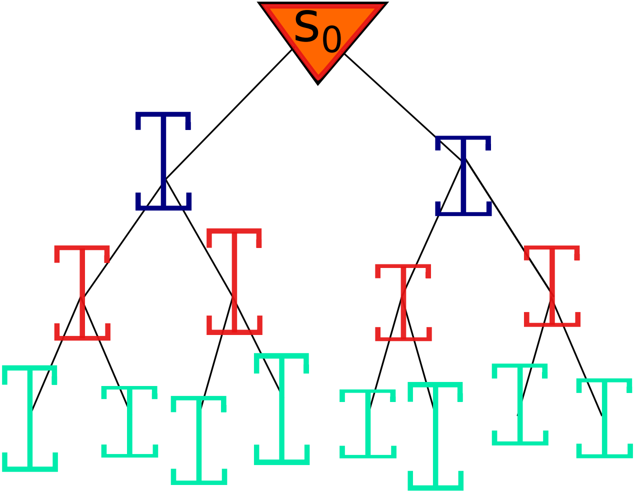
$$\mathrm{UCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{UCB}_c(t) \qquad \mathrm{LCB}_s(t) = \max_{c \in \mathcal{C}(s)} \mathrm{LCB}_c(t)$$



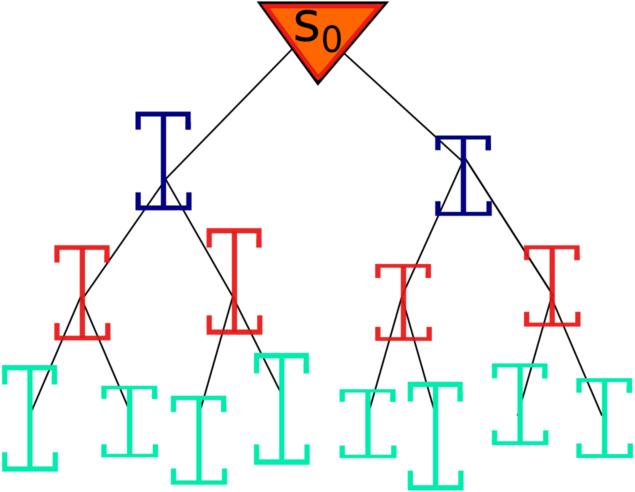
A key building block: confidence intervals

MIN node:

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



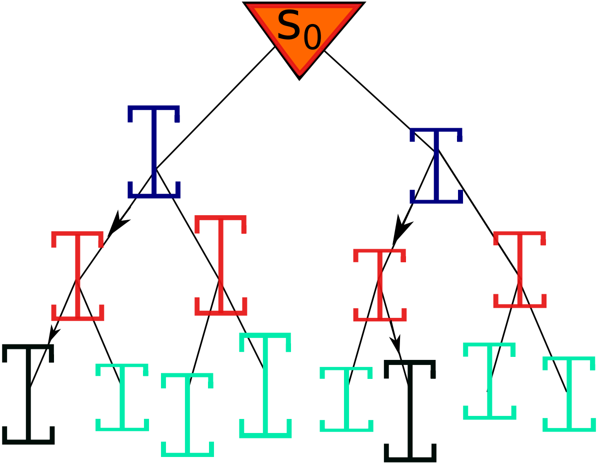
Property of this construction



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

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 $\mathrm{BAIStop}\left(\{s \in \mathcal{C}(s_0)\}\right)$ **do**

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

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

Generic BAI-MCTS algorithm

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

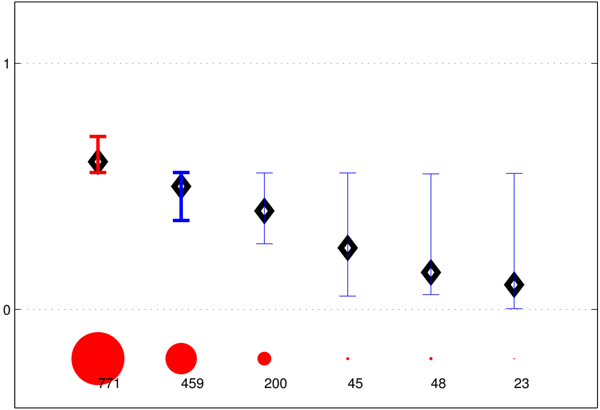
  $R_{t+1} = \mathrm{BAIStep}\left(\{s \in \mathcal{C}(s_0)\}\right)$
  Sample the representative leaf $L_{t+1} = \ell_{R_{t+1}}(t)$
  Update the information about the arms. $t = t + 1$.

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

... sometimes reduces to updating confidence intervals!

Outline

An example of BAI algorithm: LUCB



The (KL)-LUCB algorithm

[Kalyanakrishnan et al. 12, Kaufmann and Kalyanakrishnan 13]

UGapE-MCTS

based on the UGapE algorithm [Gabillon et al. 12]

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

$$\underline{a}_t = \underset{a \in \mathcal{C}(s_0)}{\text{argmin}}\ B_a(t) \quad \text{and} \quad \underline{b}_t = \underset{b \in \mathcal{C}(s_0) \setminus \{\underline{a}_t\}}{\text{argmax}}\ \mathrm{UCB}_b(t),$$

where

$$B_s(t) = \max_{s' \in \mathcal{C}(s_0) \setminus \{s\}} \mathrm{UCB}_{s'}(t) - \mathrm{LCB}_s(t).$$

- Stopping rule:

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

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

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 is (ϵ,δ)-PAC.

Theoretical guarantees

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

where

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

Sample complexity

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

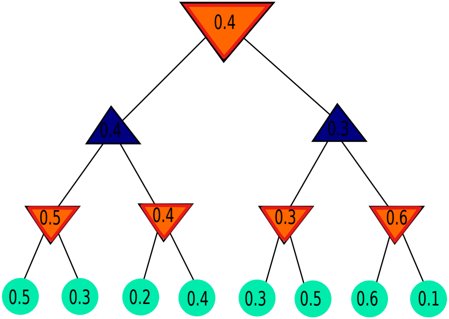
$$\tau = O\left(H^*_\epsilon(\boldsymbol{\mu}) \log\left(\frac{1}{\delta}\right)\right).$$

Theoretical guarantees

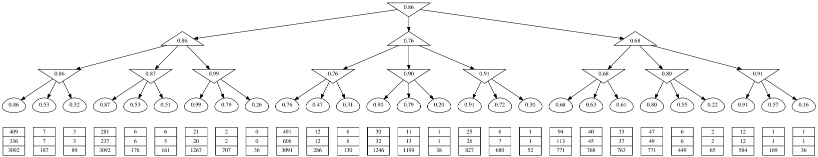
$$H_\epsilon^*(\boldsymbol{\mu}) := \sum_{\ell \in \mathcal{L}} \frac{1}{\Delta_\ell^2 \vee \Delta_*^2 \vee \epsilon^2}$$

where

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



$\epsilon = 0$, $\delta = 0.1 \cdot 27$ ($N = 10^6$ simulations)



LUCB-MCTS (0.72% errors, 1551 samples)
UGapE-MCTS (0.75% erros, 1584 samples)
FindTopWinner (0% errors, 20730 samples) [Teraoka et al. 14]

Outline

A sample complexity lower bound

Theorem

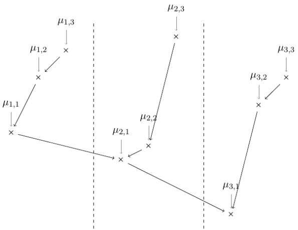
Let $\epsilon = 0$*. Any* δ*-correct algorithm satisfies*

$$\mathbb{E}_{\boldsymbol{\mu}}[\tau] \geq T^*(\boldsymbol{\mu}) \log\left(1/(3\delta)\right)$$

where

$$T^*(\boldsymbol{\mu})^{-1} := \sup_{\boldsymbol{w} \in \Sigma_{|\mathcal{L}|}} \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{\ell \in \mathcal{L}} w_\ell \mathrm{KL}\left(\mathcal{B}(\mu_\ell), \mathcal{B}(\lambda_\ell)\right).$$

Depth-two tree:



The optimal proportions satisfy

$$w^*_{i,j}(\boldsymbol{\mu}) = 0$$

if $i \geq 2$ and $j \geq 2$.

A sample complexity lower bound

Theorem

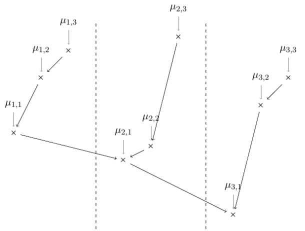
Let $\epsilon = 0$. Any δ-correct algorithm satisfies

$$\mathbb{E}_{\boldsymbol{\mu}}[\tau] \geq T^*(\boldsymbol{\mu}) \log(1/(3\delta))$$

where

$$T^*(\boldsymbol{\mu})^{-1} := \sup_{\boldsymbol{w} \in \Sigma_{|\mathcal{L}|}} \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{\ell \in \mathcal{L}} w_\ell \mathrm{KL}\left(\mathcal{B}(\mu_\ell), \mathcal{B}(\lambda_\ell)\right).$$

Depth-two tree:



The optimal proportions satisfy

$$w^*_{i,j}(\boldsymbol{\mu}) = 0$$

if $i \geq 2$ and $j \geq 2$.

A more general sparsity pattern?

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 [Ajallooeian et al., ALT 17]

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 [Ajallooeian et al., ALT 17]

Reference:

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