Reinforcement Learning Planning strategies

Odalric-Ambrym Maillard

Spring 2022

Inria Scool

Roadmap

- Monte-Carlo Tree-Search
- KL Open-Loop optimistic planning
- Best arm identification
- Hierarchical Optimistic Optimization
- Monte-Carlo Graph-Search

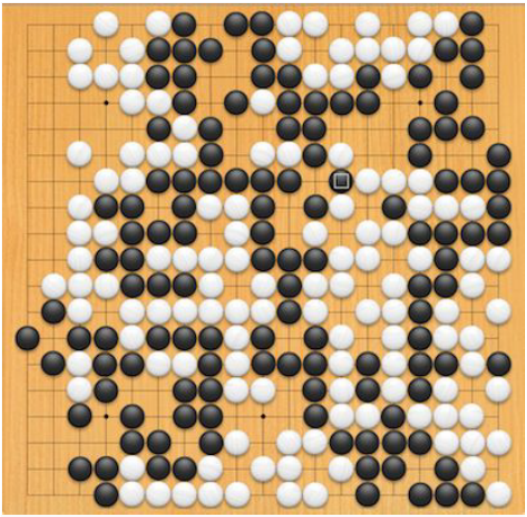
Table of contents

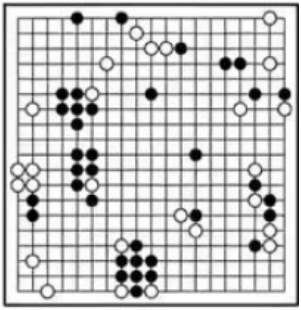
Monte-Carlo Tree Search

Planning strategy: OLOP

Best-arm identification

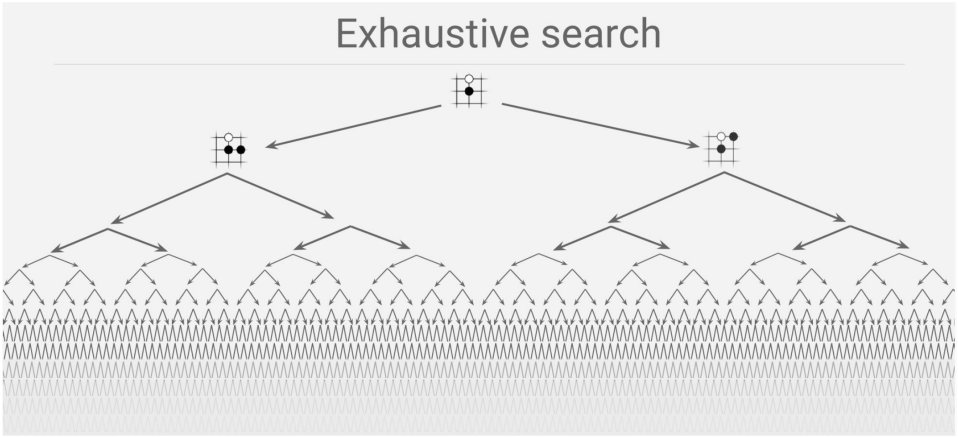
Monte-Carlo Graph Search





19×19 possible actions, 10^{100} board configurations

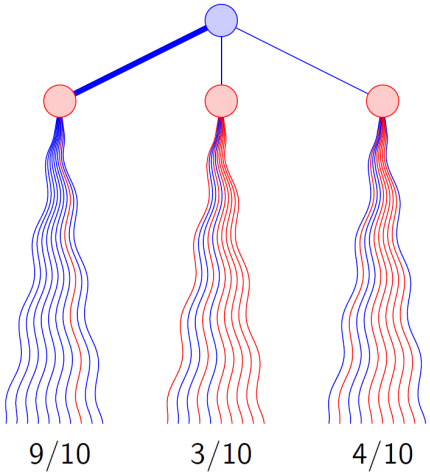
- Estimate **value** of a position ?



▷ Too large for DP, Value Iteration, or even storage. Not clear exploitable features as well.

Monte Carlo Tree Search: Pioneers

- 1993: Bernd Brügmann, but not considered seriously at that time.
- 2000-2005: the Paris School
  - Bernard Helmstetter, Tristan Cazeneuve, Bruno Bouzy, Guillaume Chaslot.
- 2006: Rémi Coulom, **MCTS** combines Monte-Carlo simulations with Tree Search.



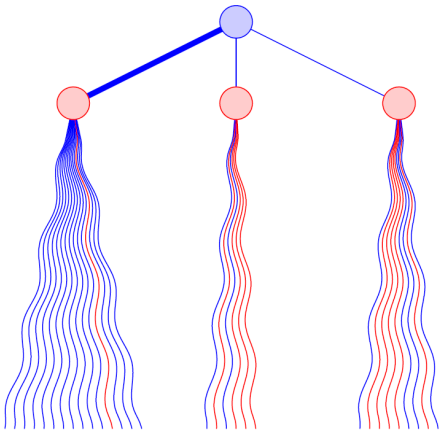
Algorithm

- N playouts for every move
- pick the best winning rate

Cost

- accurate like $1/\sqrt{N}$
- 0.01 precision requires $\sim 10,000$ playouts

Slide courtesy of R. Coulom



Idea

- more playouts to best moves

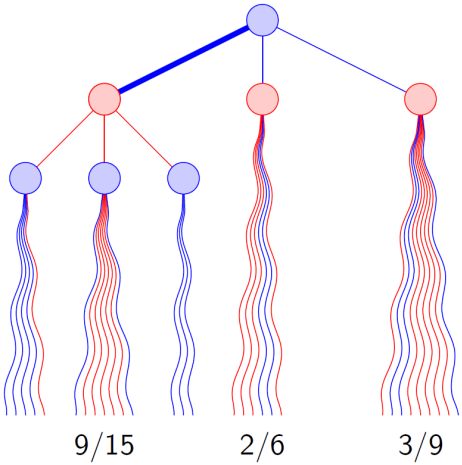
UCB: Upper Confidence Bound

$$\mathrm{UCB}_i = \frac{W_i}{N_i} + c\sqrt{\frac{\log t}{N_i}}$$

- W_i: wins (move i)
- N_i: playouts (move i)
- c: exploration parameter
- t: playouts (all moves)

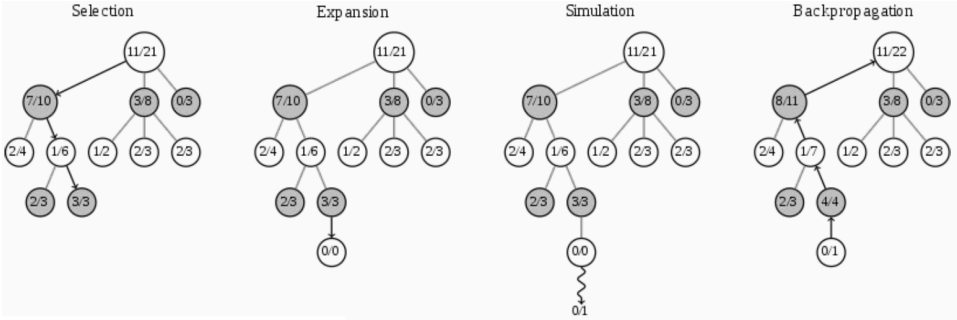
Slide courtesy of R. Coulom

Monte Carlo Tree Search: Recursive Allocation



- Apply UCB to every position visited more than N_0 times
- No min-max backup: backup average outcome
- Proved convergence to min-max value
- Best-first tree growth

Slide courtesy of R. Coulom


Selection
Expansion
Simulation
Backpropagation
11/21
7/10
3/8
0/3
2/4
1/6
1/2
2/3
2/3
2/3
3/3
0/0
0/1
11/22
8/11
1/7
4/4

▷ For generic **reward** function: At a leaf node x, we run some policy π until end of game several times. Each run gives one trajectory, with corresponding rewards

$$\widehat{v}(x) = r(x) + r(s_1) + r(s_2) + ...$$

(Monte Carlo) Average the cumulated rewards over all trajectories:

$$\widehat{V}(x) = \frac{1}{n_x} \sum_{i=1}^{n_x} \widehat{v}_i(x)$$

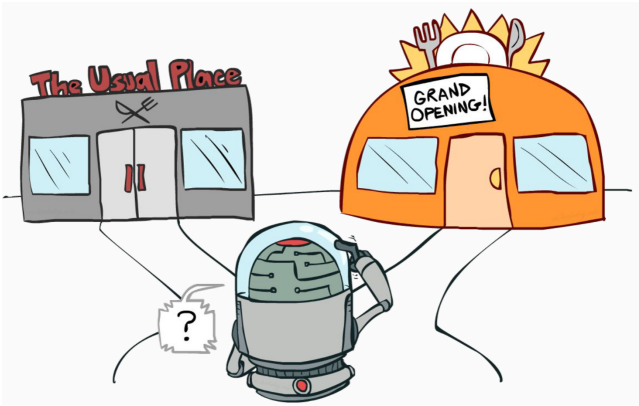
▷ **Careful**: empirical mean is only an **estimate** of true mean

Q: How much error due to n_x observations ?

A: about $\sqrt{\frac{\log(n_{x-1})}{n_x}}$ where $x-1$ denotes parent node of x.

UCT exploit explore

$$x^* = arg\max_x \left(\frac{w_x}{n_x} + c\sqrt{\frac{\log n_{x-1}}{n_x}} \right)$$


The Usual Place
GRAND OPENING!
?

MCTS successes in Go

- 2006: Gold Turin in Olympiad on 9 x 9
- Nowadays: Alpha-star, Alpha-zero, etc. are all based on MCTS.

Monte-Carlo Tree Search

Planning strategy: OLOP

Best-arm identification

Monte-Carlo Graph Search

- Planning is not solving cumulative regret, but what is called **simple regret**:

$$\text{Minimize } \mathbb{E}[V^\star - V_n]$$

where V_n is value estimated after having queried model n times. n is budget.

- When n is larger (more simulations allowed, more time to think), better accuracy.

Practical Open-Loop Optimistic Planning

Edouard Leurent[1,2], **Odalric-Ambrym Maillard**[1]

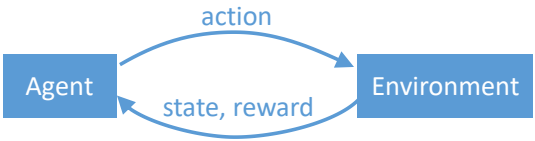
[1] SequeL, Inria Lille – Nord Europe

[2] Renault Group

ECML PKDD 2019

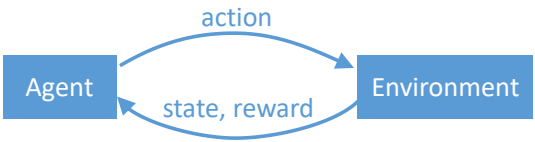
Würzburg, September 2019

Motivation — Sequential Decision Making



Markov Decision Processes

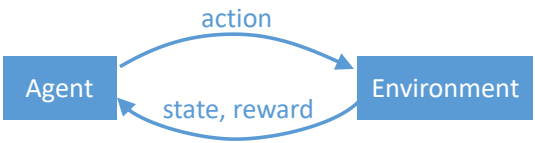
Motivation — Sequential Decision Making



Markov Decision Processes

1. Observe state $s \in S$;

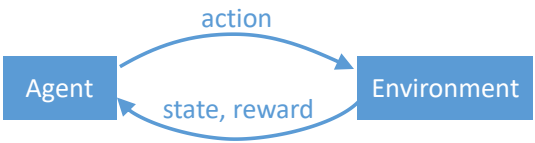
Motivation — Sequential Decision Making



Markov Decision Processes

1. Observe state $s \in S$;
2. Pick a discrete action $a \in A$;

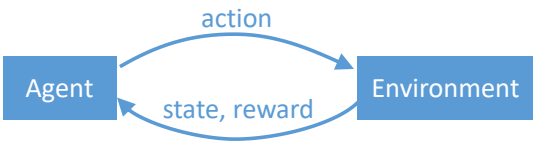
Motivation — Sequential Decision Making



Markov Decision Processes

1. Observe state $s \in S$;
2. Pick a discrete action $a \in A$;
3. Transition to a next state $s' \sim \mathbb{P}\left(s'|s,a\right)$;

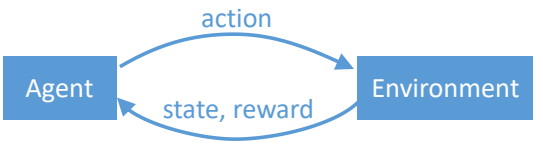
Motivation — Sequential Decision Making



Markov Decision Processes

1. Observe state $s \in S$;
2. Pick a discrete action $a \in A$;
3. Transition to a next state $s' \sim \mathbb{P}(s'|s,a)$;
4. Receive a bounded reward $r \in [0,1]$ drawn from $\mathbb{P}(r|s,a)$.

Motivation — Sequential Decision Making



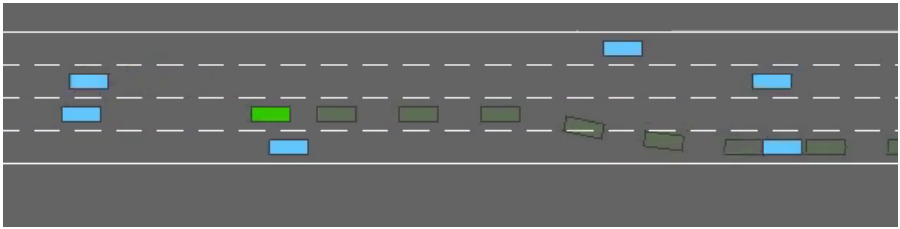
Markov Decision Processes

1. Observe state $s \in S$;
2. Pick a discrete action $a \in A$;
3. Transition to a next state $s' \sim \mathbb{P}(s'|s,a)$;
4. Receive a bounded reward $r \in [0,1]$ drawn from $\mathbb{P}(r|s,a)$.

Objective: maximise $\mathrm{V} = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t r_t\right]$

Motivation — Example

The `highway-env` environment

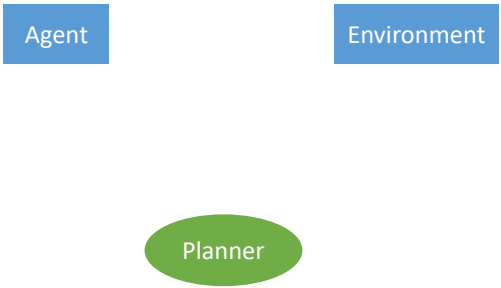


We want to handle stochasticity.

Motivation — How to solve MDPs?

Online *Planning*

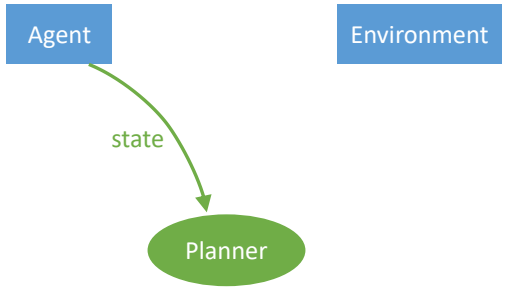
- we have access to a generative model:
  - ↳ yields samples of $s', r \sim \mathbb{P}(s', r|s, a)$ when queried



Motivation — How to solve MDPs?

Online *Planning*

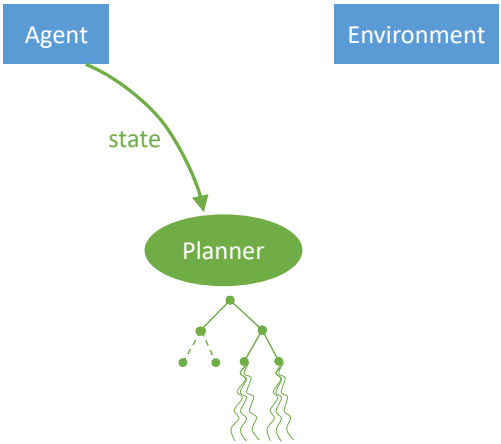
- we have access to a generative model:
  - ↳ yields samples of $s', r \sim \mathbb{P}(s', r|s, a)$ when queried



Motivation — How to solve MDPs?

Online *Planning*

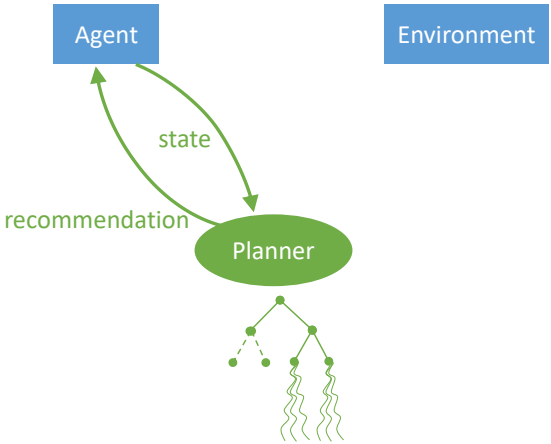
- we have access to a generative model:
  - ↳ yields samples of $s', r \sim \mathbb{P}(s', r|s, a)$ when queried



Motivation — How to solve MDPs?

Online *Planning*

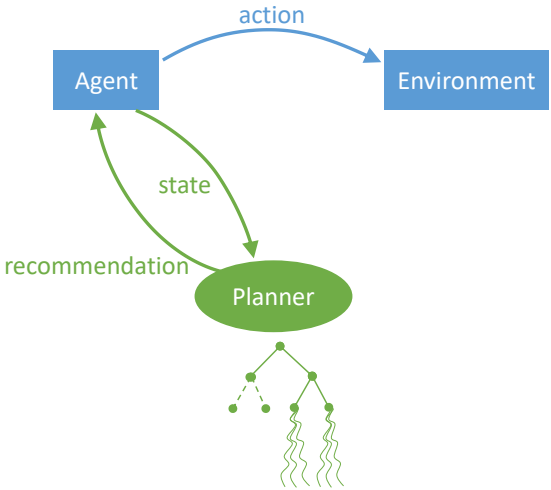
- we have access to a generative model:
  - ↳ yields samples of $s', r \sim \mathbb{P}(s', r|s, a)$ when queried



Motivation — How to solve MDPs?

Online *Planning*

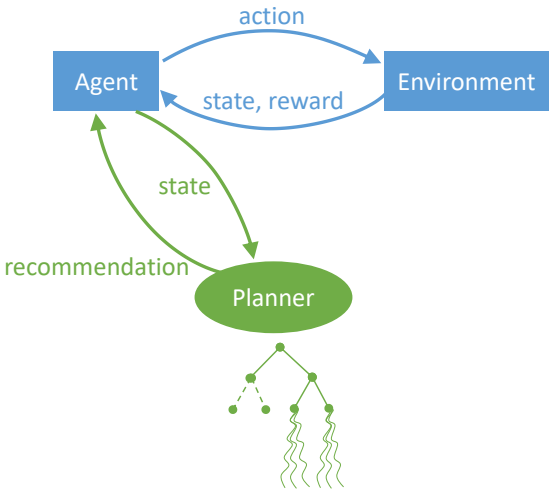
- we have access to a generative model:
  - ↳ yields samples of $s', r \sim \mathbb{P}(s', r|s, a)$ when queried



Motivation — How to solve MDPs?

Online *Planning*

- we have access to a generative model:
  - ↳ yields samples of $s', r \sim \mathbb{P}(s', r|s, a)$ when queried



Motivation — How to solve MDPs?

Online *Planning*

- fixed budget: the model can only be queried n times

Objective: minimize $\mathbb{E}\ \underbrace{V^* - V(n)}_{\text{Simple Regret } r_n}$

An exploration-exploitation problem.

Optimistic Planning

Optimism in the Face of Uncertainty

Given a set of options $a \in A$ with uncertain outcomes, try the one with the highest possible outcome.

Optimistic Planning

Optimism in the Face of Uncertainty

Given a set of options $a \in A$ with uncertain outcomes, try the one with the highest possible outcome.

- Either you performed well;

Optimistic Planning

Optimism in the Face of Uncertainty

Given a set of options $a \in A$ with uncertain outcomes, try the one with the highest possible outcome.

- Either you performed well;
- or you learned something.

Optimistic Planning

Optimism in the Face of Uncertainty

Given a set of options $a \in A$ with uncertain outcomes, try the one with the highest possible outcome.

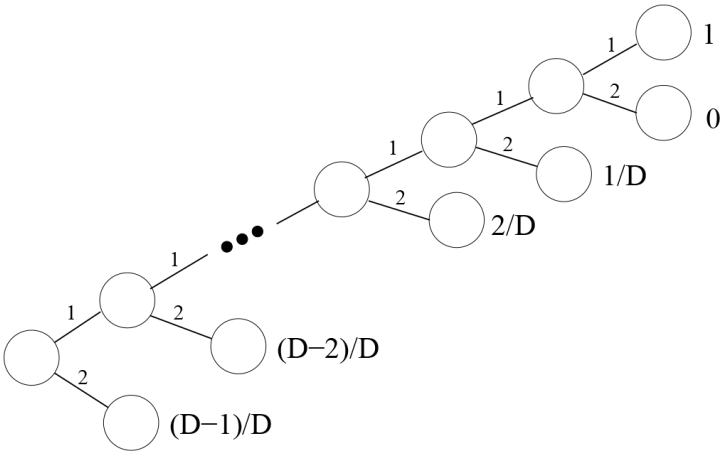
- Either you performed well;
- or you learned something.

Instances

- Monte-carlo tree search (`MCTS`) [Coulom 2006]: `CrazyStone`
- Reframed in the bandit setting as `UCT` [Kocsis and Szepesvári 2006], still very popular (e.g. `Alpha Go`).
- Proved asymptotic consistency, but no regret bound.

Analysis of UCT

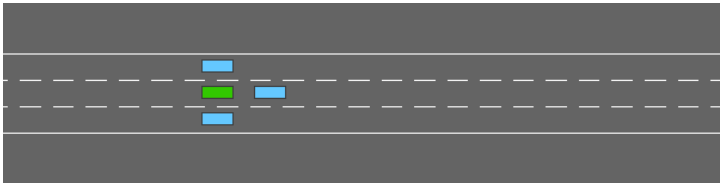
It was analysed in [Coquelin and Munos 2007]

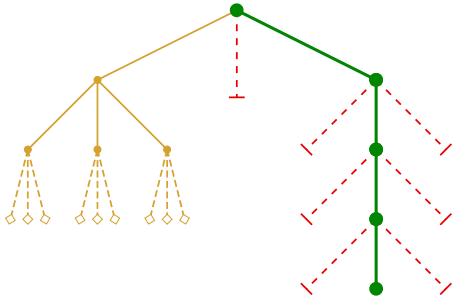


The sample complexity of is lower-bounded by $O(\exp(\exp(D)))$.

Failing cases of UCT

Not just a theoretical counter-example.





Can we get better guarantees?

OPD: Optimistic Planning for Deterministic systems

- Introduced by [Hren and Munos 2008]
- Another optimistic algorithm
- Only for deterministic MDPs

Theorem (OPD sample complexity)

$$\mathbb{E}\, r_n = \mathcal{O}\left(n^{-\frac{\log 1/\gamma}{\log \kappa}}\right), \text{if } \kappa > 1$$

Can we get better guarantees?

OPD: Optimistic Planning for Deterministic systems

- Introduced by [Hren and Munos 2008]
- Another optimistic algorithm
- Only for deterministic MDPs

Theorem (OPD sample complexity)

$$\mathbb{E}\, r_n = \mathcal{O}\left(n^{-\frac{\log 1/\gamma}{\log \kappa}}\right), \text{if } \kappa > 1$$

OLOP: Open-Loop Optimistic Planning

- Introduced by [Bubeck and Munos 2010]
- Extends OPD to the stochastic setting
- Only considers open-loop policies, i.e. sequences of actions

The idea behind OLOP

A direct application of Optimism in the Face of Uncertainty

1. We want

$$\max_a V(a)$$

The idea behind OLOP

A direct application of Optimism in the Face of Uncertainty

1. We want
$$\max_a V(a)$$

2. Form upper confidence-bounds of sequence values:
$$V(a) \leq U_a \quad \text{w.h.p}$$

The idea behind OLOP

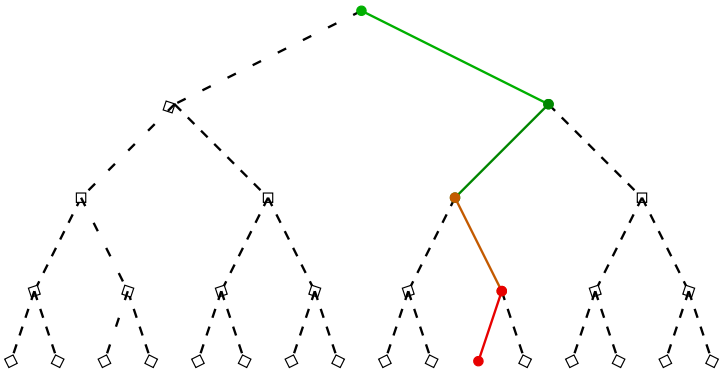
A direct application of Optimism in the Face of Uncertainty

1. We want
$$\max_a V(a)$$

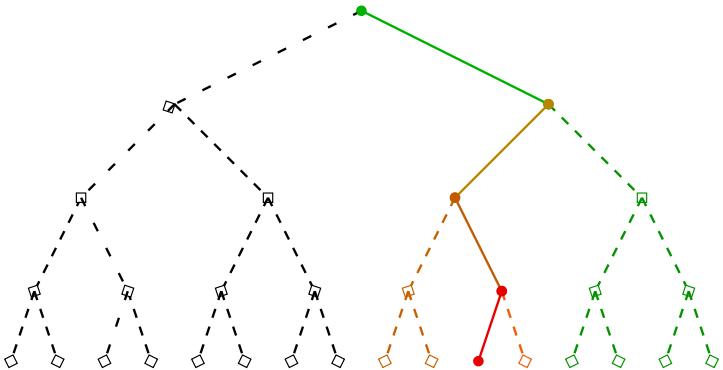
2. Form upper confidence-bounds of sequence values:
$$V(a) \leq U_a \quad \text{w.h.p}$$

3. Sample the sequence with highest UCB:
$$\arg\max_a U_a$$

The idea behind OLOP



The idea behind OLOP



Under the hood

Upper-bounding the value of sequences

$$V(a) = \overbrace{\sum_{t=1}^{h} \gamma^t \mu_{a_{1:t}}}^{\text{follow the sequence}} + \overbrace{\sum_{t \geq h+1} \gamma^t \mu_{a^*_{1:t}}}^{\text{act optimally}}$$

Under the hood

Upper-bounding the value of sequences

$$V(a) = \overbrace{\sum_{t=1}^{h} \gamma^t \underbrace{\mu_{a_{1:t}}}_{\leq U^\mu}}^{\text{follow the sequence}} + \overbrace{\sum_{t \geq h+1} \gamma^t \underbrace{\mu_{a^*_{1:t}}}_{\leq 1}}^{\text{act optimally}}$$

Under the hood

OLOP main tool: the Chernoff-Hoeffding deviation inequality

$$\underbrace{U_a^{\mu}(m)}_{\text{Upper bound}} \overset{\text{def}}{=} \underbrace{\hat{\mu}_a(m)}_{\text{Empirical mean}} + \underbrace{\sqrt{\frac{2\log M}{T_a(m)}}}_{\text{Confidence interval}}$$

Under the hood

OLOP main tool: the Chernoff-Hoeffding deviation inequality

$$\underbrace{U_a^{\mu}(m)}_{\text{Upper bound}} \overset{\text{def}}{=} \underbrace{\hat{\mu}_a(m)}_{\text{Empirical mean}} + \underbrace{\sqrt{\frac{2\log M}{T_a(m)}}}_{\text{Confidence interval}}$$

OPD: upper-bound all the future rewards by 1

$$U_a(m) \overset{\text{def}}{=} \sum_{t=1}^{h} \underbrace{\gamma^t U_{a_{1:t}}^{\mu}(m)}_{\text{Past rewards}} + \underbrace{\frac{\gamma^{h+1}}{1-\gamma}}_{\text{Future rewards}}$$

Under the hood

OLOP main tool: the Chernoff-Hoeffding deviation inequality

$$\underbrace{U_a^{\mu}(m)}_{\text{Upper bound}} \stackrel{\text{def}}{=} \underbrace{\hat{\mu}_a(m)}_{\text{Empirical mean}} + \underbrace{\sqrt{\frac{2\log M}{T_a(m)}}}_{\text{Confidence interval}}$$

OPD: upper-bound all the future rewards by 1

$$U_a(m) \stackrel{\text{def}}{=} \sum_{t=1}^{h} \underbrace{\gamma^t U_{a_{1:t}}^{\mu}(m)}_{\text{Past rewards}} + \underbrace{\frac{\gamma^{h+1}}{1-\gamma}}_{\text{Future rewards}}$$

Bounds sharpening

$$B_a(m) \stackrel{\text{def}}{=} \inf_{1 \le t \le L} U_{a_{1:t}}(m)$$

OLOP guarantees

Theorem (OLOP Sample complexity)

OLOP satisfies:

$$\mathbb{E}\, r_n = \begin{cases} \widetilde{\mathcal{O}}\left(n^{-\frac{\log 1/\gamma}{\log \kappa'}}\right), & \text{if } \gamma\sqrt{\kappa'} > 1 \\ \widetilde{\mathcal{O}}\left(n^{-\frac{1}{2}}\right), & \text{if } \gamma\sqrt{\kappa'} \leq 1 \end{cases}$$

"Remarkably, in the case $\kappa\gamma^2 > 1$, we obtain the same rate for the simple regret as Hren and Munos (2008). Thus, in this case, we can say that planning in stochastic environments is not harder than planning in deterministic environments".

Does it work?

Our objective: understand and bridge this gap.

Make OLOP *practical*.

What's wrong with OLOP?

Explanation: inconsistency

- Unintended behaviour happens when $U_a^\mu(m) > 1, \forall a$.

$$U_a^\mu(m) = \underbrace{\hat{\mu}_a(m)}_{\in[0,1]} + \underbrace{\sqrt{\frac{2\log M}{T_a(m)}}}_{>0}$$

What's wrong with OLOP?

Explanation: inconsistency

- Unintended behaviour happens when $U_a^\mu(m) > 1, \forall a$.

$$U_a^\mu(m) = \underbrace{\hat{\mu}_a(m)}_{\in[0,1]} + \underbrace{\sqrt{\frac{2\log M}{T_a(m)}}}_{>0}$$

- Then the sequence $(U_{a_{1:t}}(m))_t$ is increasing

$$U_{a_{1:1}}(m) = \gamma U_{a_1}^\mu(m) + \gamma^2 1 \qquad +\gamma^3 1 + \dots$$
$$U_{a_{1:2}}(m) = \gamma U_{a_1}^\mu(m) + \gamma^2 \underbrace{U_{a_2}^\mu}_{>1} \qquad +\gamma^3 1 + \dots$$

What's wrong with OLOP?

Explanation: inconsistency

- Unintended behaviour happens when $U_a^\mu(m) > 1, \forall a$.

$$U_a^\mu(m) = \underbrace{\hat{\mu}_a(m)}_{\in[0,1]} + \underbrace{\sqrt{\frac{2\log M}{T_a(m)}}}_{>0}$$

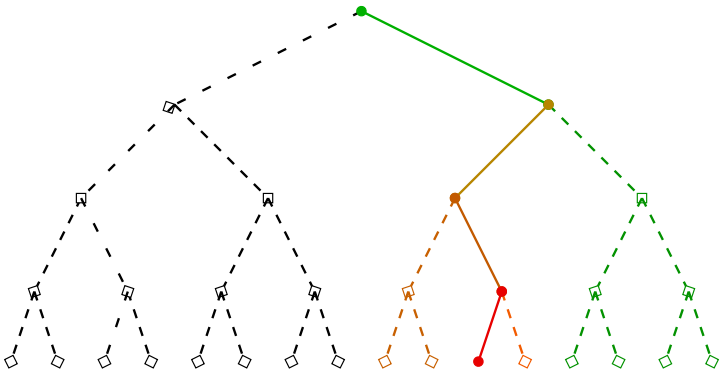
- Then the sequence $(U_{a_{1:t}}(m))_t$ is increasing

$$\begin{aligned} U_{a_{1:1}}(m) &= \gamma U_{a_1}^\mu(m) + \gamma^2 1 && +\gamma^3 1 + \dots \\ U_{a_{1:2}}(m) &= \gamma U_{a_1}^\mu(m) + \gamma^2 \underbrace{U_{a_2}^\mu}_{>1} && +\gamma^3 1 + \dots \end{aligned}$$

- Then $B_a(m) = U_{a_{1:1}}(m)$

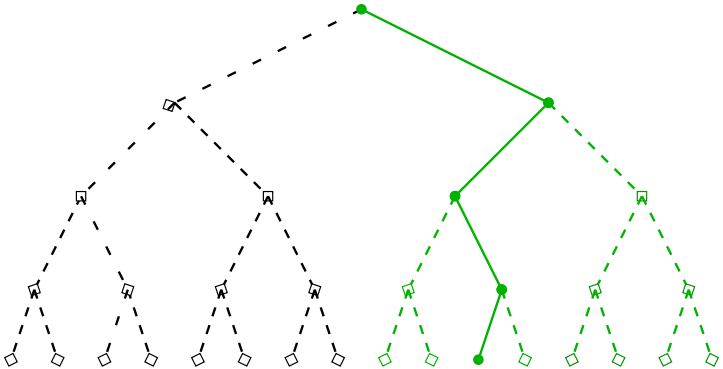
What's wrong with OLOP?

What we were promised



What's wrong with OLOP?

What we actually get



OLOP behaves as uniform planning!

Our contribution: Kullback-Leibler OLOP

We summon the upper-confidence bound from kl-UCB [Cappé et al. 2013]:

$$U_a^{\mu}(m) \stackrel{\text{def}}{=} \max\{q \in I : T_a(m) d(\hat{\mu}_a(m), q) \leq f(m)\}$$

Our contribution: Kullback-Leibler OLOP

We summon the upper-confidence bound from kl-UCB [Cappé et al. 2013]:

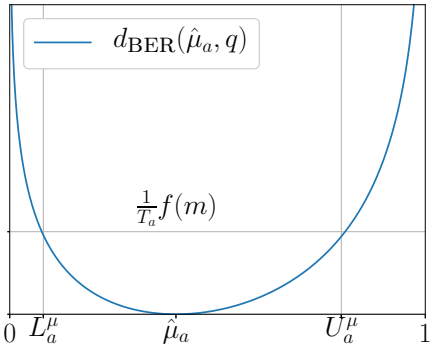
$$U_a^{\mu}(m) \stackrel{\text{def}}{=} \max\{q \in I : T_a(m) d(\hat{\mu}_a(m), q) \leq f(m)\}$$

Algorithm	OLOP	KL-OLOP
Interval I	$\mathbb{R}$	$[0, 1]$
Divergence d	d_{QUAD}	d_{BER}
$f(m)$	$4 \log M$	$2 \log M + 2 \log\log M$

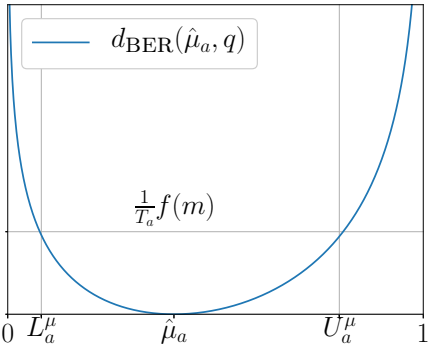
$$d_{\text{QUAD}}(p, q) \stackrel{\text{def}}{=} 2(p - q)^2$$

$$d_{\text{BER}}(p, q) \stackrel{\text{def}}{=} p \log \frac{p}{q} + (1 - p) \log \frac{1 - p}{1 - q}$$

Our contribution: Kullback-Leibler OLOP



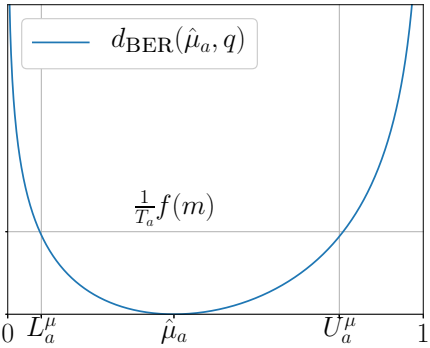
Our contribution: Kullback-Leibler OLOP



And now,

- $U_a^\mu(m) \in I = [0,1], \forall a$.

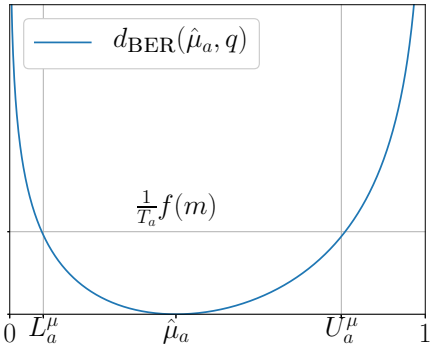
Our contribution: Kullback-Leibler OLOP



And now,

- $U_a^\mu(m) \in I = [0,1], \forall a$.
- The sequence $(U_{a_{1:t}}(m))_t$ is non-increasing

Our contribution: Kullback-Leibler OLOP



And now,

- $U_a^\mu(m) \in I = [0,1], \forall a$.
- The sequence $(U_{a_{1:t}}(m))_t$ is non-increasing
- $B_a(m) = U_a(m)$, the *bound sharpening* step is superfluous.

Sample complexity

Theorem (Sample complexity)

KL-OLOP enjoys the same regret bounds as OLOP. More precisely, KL-OLOP satisfies:

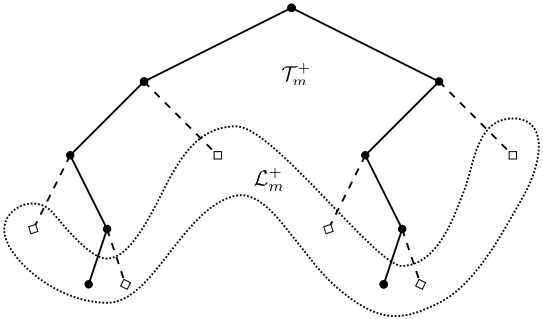
$$\mathbb{E}\, r_n = \begin{cases} \widetilde{\mathcal{O}}\left(n^{-\frac{\log 1/\gamma}{\log \kappa'}}\right), & \text{if } \gamma\sqrt{\kappa'} > 1 \\ \widetilde{\mathcal{O}}\left(n^{-\frac{1}{2}}\right), & \text{if } \gamma\sqrt{\kappa'} \leq 1 \end{cases}$$

Time complexity

Original KL-OLOP

Compute $B_a(m-1)$ *from* (14) *for all* $a \in A^L$

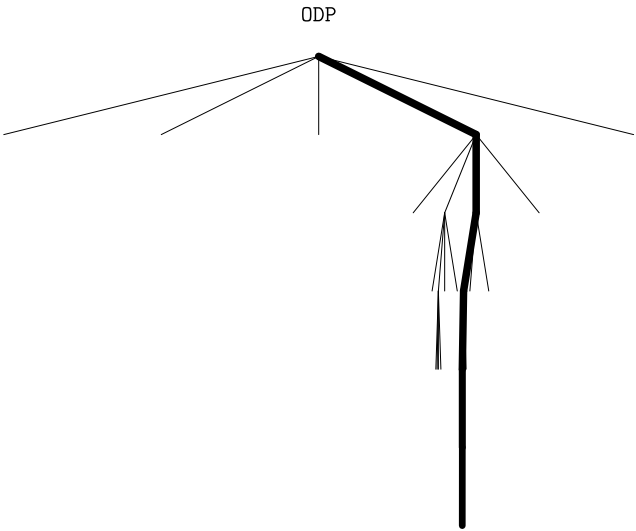
Lazy KL-OLOP



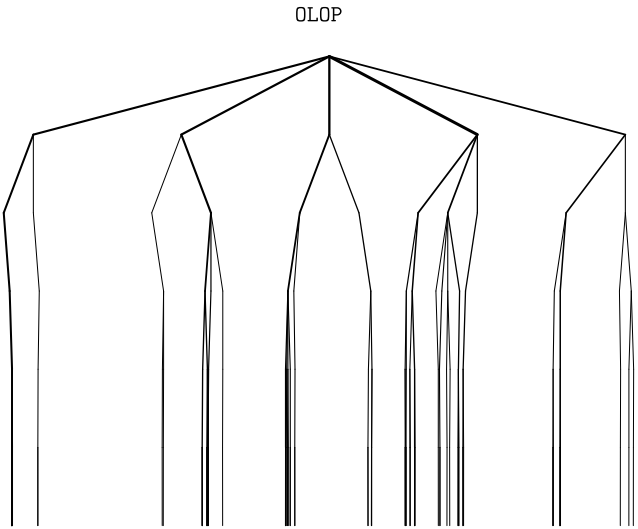
Property (Time and memory complexity)

$$\frac{C(\texttt{Lazy KL-OLOP})}{C(\texttt{KL-OLOP})} = \frac{nK}{K^L}$$

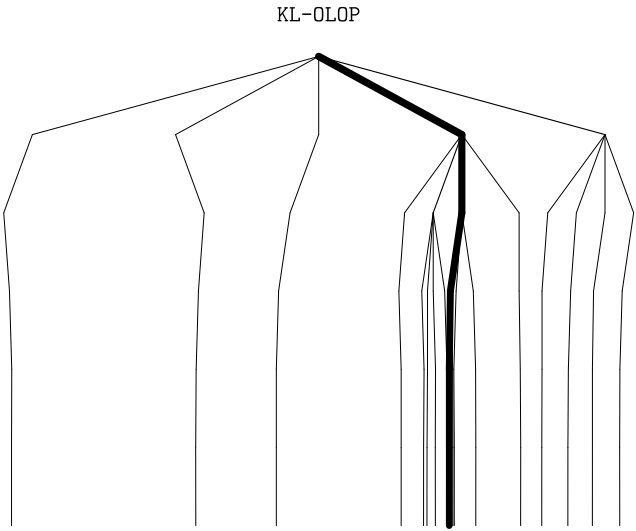
Experiments — Expanded Trees



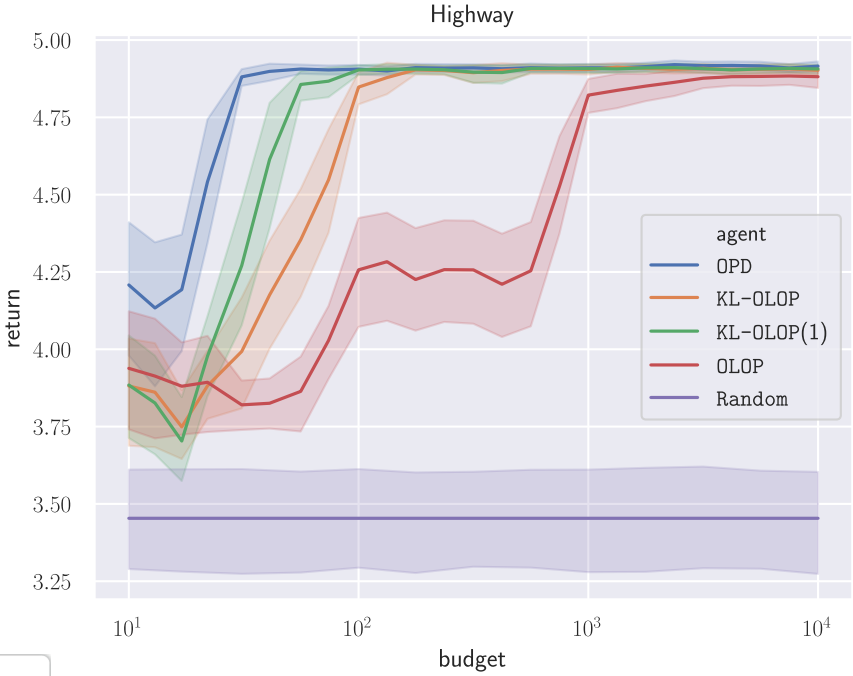
Experiments — Expanded Trees



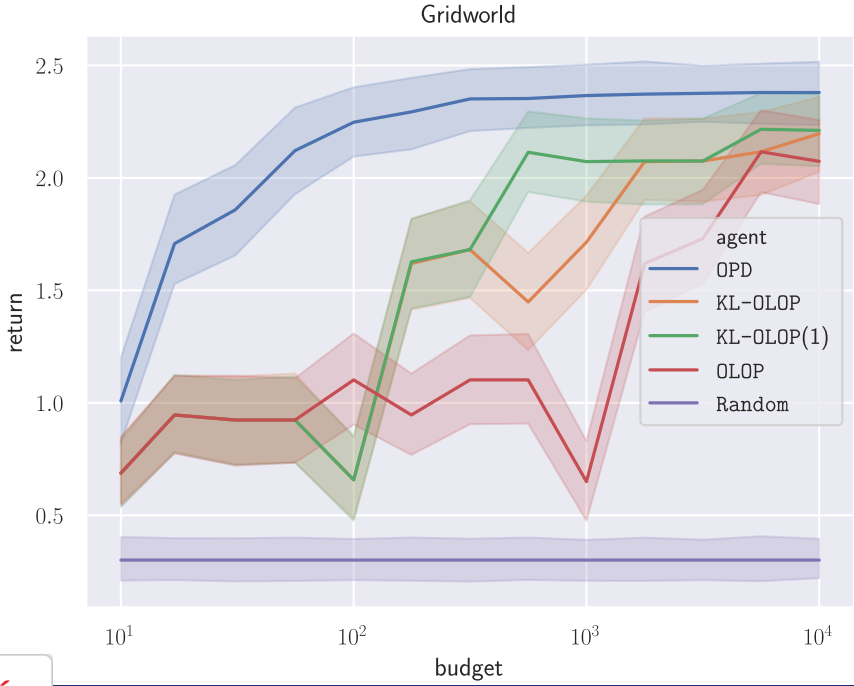
Experiments — Expanded Trees



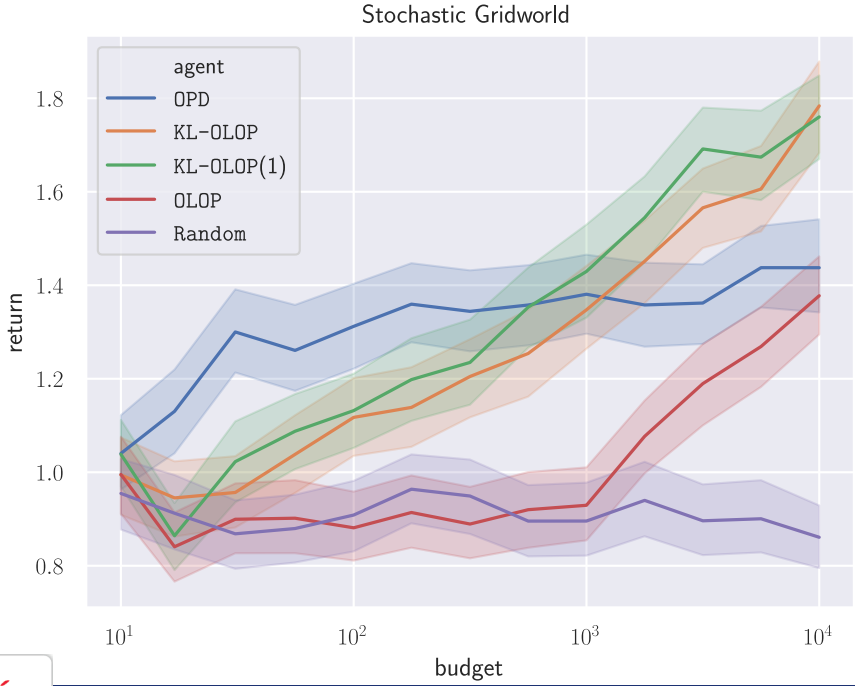
Experiments — Highway



Experiments — Gridworld



Experiments — Stochastic Gridworld



References


Sébastien Bubeck and Rémi Munos. "Open Loop Optimistic Planning". In: *Proc. of COLT*. 2010.


Olivier Cappé, Aurélien Garivier, Odalric-Ambrym Maillard, Rémi Munos, and Gilles Stoltz. "Kullback-Leibler Upper Confidence Bounds for Optimal Sequential Allocation". In: *The Annals of Statistics* 41.3 (2013), pp. 1516–1541.


Pierre-Arnaud Coquelin and Rémi Munos. "Bandit Algorithms for Tree Search". In: *Proc. of UAI* (2007).


Rémi Coulom. "Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search". In: *Proc. of International Conference on Computer and Games*. 2006.


Jean François Hren and Rémi Munos. "Optimistic planning of deterministic systems". In: *Lecture Notes in Computer Science* (2008).


Levente Kocsis and Csaba Szepesvári. "Bandit Based Monte-carlo Planning". In: *Proc. of ECML PKDD*. 2006.

Monte-Carlo Tree Search

Planning strategy: OLOP

Best-arm identification

Monte-Carlo Graph Search

Let us go back to bandit.

The Stochastic Multi-Armed Bandit Setup

K **arms** $\leftrightarrow$ 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
- receives a reward $R_t = X_{A_t,t} \sim \nu_{A_t}$

Sequential sampling strategy (**bandit algorithm**):

$$A_{t+1} = F_t(A_1, R_1, \ldots, A_t, R_t).$$

Goal: Maximize $\mathbb{E}\left[\sum_{t=1}^{T} R_t\right]$

The Stochastic Multi-Armed Bandit Setup

K **arms** $\leftrightarrow$ 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
- receives a sample $X_t = X_{A_t,t} \sim \nu_{A_t}$

Sequential sampling strategy (**bandit algorithm**):

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

Goal: Maximize $\mathbb{E}\left[\sum_{t=1}^{T} X_t\right]$ $\rightarrow$ not the only possible goal!

Should we maximize rewards?





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

For the t-th patient in a clinical study,

- chooses a treatment A_t
- observes a response $X_t \in \{0, 1\}$: $\mathbb{P}(X_t = 1) = \mu_{A_t}$

Maximize rewards $\leftrightarrow$ cure as many patients as possible

Should we maximize rewards?





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

For the t-th patient in a clinical study,

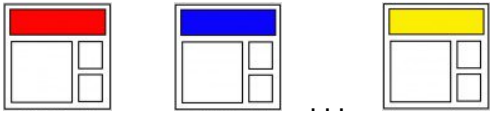
- chooses a treatment A_t
- observes a response $X_t \in \{0, 1\}$: $\mathbb{P}(X_t = 1) = \mu_{A_t}$

Maximize rewards $\leftrightarrow$ cure as many patients as possible

Alternative goal: identify as quickly as possible the best treatment (without trying to cure patients during the study)

Should we maximize rewards?

Probability that some version of a website generates a conversion:



μ_1 μ_2 μ_K

Best version: $a_\star = \underset{a=1,\dots,K}{\operatorname{argmax}}\ \mu_a$

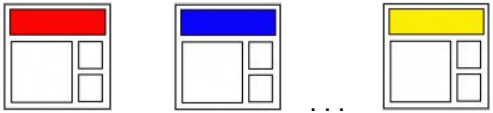
Sequential protocol: for the t-th visitor:

- display version A_t
- observe conversion indicator $X_t \sim \mathcal{B}(\mu_{A_t})$.

Maximize rewards $\leftrightarrow$ maximize the number of conversions

Should we maximize rewards?

Probability that some version of a website generates a conversion:



μ_1 μ_2 μ_K

Best version: $a_\star = \underset{a=1,\dots,K}{\operatorname{argmax}}\ \mu_a$

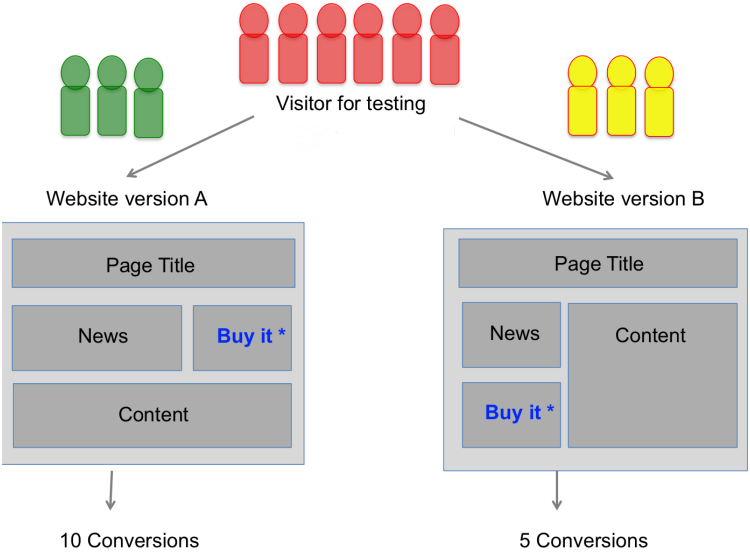
Sequential protocol: for the t-th visitor:

- display version A_t
- observe conversion indicator $X_t \sim \mathcal{B}(\mu_{A_t})$.

Maximize rewards $\leftrightarrow$ maximize the number of conversions

Alternative goal: identify the best version
(without trying to maximize conversions during the test)

A/B Testing ($K = 2$)



Goal: find the best version.

Improve your A/B tests?

A way to do A/B Testing:

- allocate n_A users to page A and n_B users to page B (decided in advance, often $n_A = n_B$)
- perform a statistical test of "A better than B"

Improve your A/B tests?

A way to do A/B Testing:

- allocate n_A users to page A and n_B users to page B (decided in advance, often $n_A = n_B$)
- perform a statistical test of "A better than B"

Alternative: fully adaptive A/B Testing

- sequentially choose which version to allocate to each visitor
- (adaptively choose when to stop the experiment)

➜ best arm identification in a bandit model

Finding the best arm in a bandit model

Pure Exploration in Bandit Models

Goal: identify an arm with large mean as quickly and accurately as possible $\simeq$ identify

$$a_\star = \underset{a=1,\dots,K}{\operatorname{argmax}} \ \mu_a.$$

Algorithm: made of three components:

- sampling rule: A_t (arm to explore)
- recommendation rule: B_t (current guess for the best arm)
- stopping rule τ (when do we stop exploring?)

Probability of error [Even-Dar et al., 2006, Audibert et al., 2010]

The probability of error after n rounds is

$$p_\nu(n) = \mathbb{P}_\nu\left(B_n \neq a_\star\right).$$

Pure Exploration in Bandit Models

Goal: identify an arm with large mean as quickly and accurately as possible $\simeq$ identify

$$a_\star = \underset{a=1,\dots,K}{\operatorname{argmax}} \ \mu_a.$$

Algorithm: made of three components:

- sampling rule: A_t (arm to explore)
- recommendation rule: B_t (current guess for the best arm)
- stopping rule τ (when do we stop exploring?)

Simple regret [Bubeck et al., 2009]

The simple regret after n rounds is

$$r_\nu(n) = \mu_\star - \mu_{B_n}.$$

Pure Exploration in Bandit Models

Goal: identify an arm with large mean as quickly and accurately as possible $\simeq$ identify

$$a_\star = \underset{a=1,\dots,K}{\operatorname{argmax}} \ \mu_a.$$

Algorithm: made of three components:

- ➜ sampling rule: A_t (arm to explore)
- ➜ recommendation rule: B_t (current guess for the best arm)
- ➜ stopping rule τ (when do we stop exploring?)

Simple regret [Bubeck et al., 2009]

The simple regret after n rounds is

$$r_\nu(n) = \mu_\star - \mu_{B_n}.$$

$$\Delta_{\min} p_\nu(n) \leq \mathbb{E}_\nu[r_\nu(n)] \leq \Delta_{\max} p_\nu(n)$$

Several objectives

Algorithm: made of three components:

- ➜ sampling rule: A_t (arm to explore)
- ➜ recommendation rule: B_t (current guess for the best arm)
- ➜ stopping rule τ (when do we stop exploring?)

- **Objectives studied in the literature:**

Fixed-confidence setting	Fixed-budget setting	Anytime exploration
input: risk parameter δ (tolerance parameter ϵ)	input: budget T	no input
minimize $\mathbb{E}[\tau]$ $\mathbb{P}(B_\tau \neq a_\star) \leq \delta$ or $\mathbb{P}(r_\nu(\tau) < \epsilon) \leq \delta$	$\tau = T$ minimize $\mathbb{P}(B_T \neq a_\star)$ or $\mathbb{E}[r_T(\nu)]$	for all t, minimize $p_\nu(t)$ or $\mathbb{E}[r_\nu(t)]$
[Even-Dar et al., 2006]	[Bubeck et al., 2009] [Audibert et al., 2010]	[Jun and Nowak, 2016]

Outline

Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

Can we use UCB?

Context: bounded rewards (ν_a supported in $[0,1]$)

We know good algorithms to maximize rewards, for example UCB(α)

$$A_{t+1} = \underset{a=1,\dots,K}{\operatorname{argmax}} \ \hat{\mu}_a(t) + \sqrt{\alpha \frac{\ln(t)}{N_a(t)}}$$

- How good is it for best arm identification?

Can we use UCB?

Context: bounded rewards (ν_a supported in $[0,1]$)

We know good algorithms to maximize rewards, for example UCB(α)

$$A_{t+1} = \underset{a=1,\dots,K}{\operatorname{argmax}} \ \hat{\mu}_a(t) + \sqrt{\alpha \frac{\ln(t)}{N_a(t)}}$$

- How good is it for best arm identification?

Possible recommendation rules:

Empirical Best Arm (EBA)	$B_t = \operatorname{argmax}_a \ \hat{\mu}_a(t)$
Most Played Arm (MPA)	$B_t = \operatorname{argmax}_a \ N_a(t)$
Empirical Distribution of Plays (EDP)	$B_t \sim p_t$, where $p_t = \left(\frac{N_1(t)}{t}, \dots, \frac{N_K(t)}{t}\right)$

[Bubeck et al., 2009]

Can we use UCB?

- **UCB + Empirical Distribution of Plays**

$$\begin{aligned}
\mathbb{E}\left[r_\nu(n)\right] &= \mathbb{E}\left[\mu_\star - \mu_{B_n}\right] = \mathbb{E}\left[\sum_{b=1}^{K}(\mu_\star - \mu_b)\mathbb{1}_{(B_n=b)}\right] \\
&= \mathbb{E}\left[\sum_{b=1}^{K}(\mu_\star - \mu_b)\mathbb{P}(B_n = b|\mathcal{F}_n)\right] \\
&= \mathbb{E}\left[\sum_{b=1}^{K}(\mu_\star - \mu_b)\frac{N_b(n)}{n}\right] \\
&= \frac{1}{n}\sum_{b=1}^{K}(\mu_\star - \mu_b)\mathbb{E}[N_b(n)] \\
&= \frac{\mathcal{R}_\nu(n)}{n}.
\end{aligned}$$

➜ a conversion from cumulated regret to simple regret!

Can we use UCB?

- **UCB + Empirical Distribution of Plays**

$$\mathbb{E}\left[r_\nu\left(\texttt{UCB}(\alpha), n\right)\right] \leq \frac{\mathcal{R}_\nu(\texttt{UCB}(\alpha), n)}{n} \leq \frac{C(\nu)\ln(n)}{n}$$

Can we use UCB?

- **UCB + Empirical Distribution of Plays**

$$\mathbb{E}\left[r_\nu\left(\texttt{UCB}(\alpha), n\right)\right] \leq \frac{\mathcal{R}_\nu(\texttt{UCB}(\alpha), n)}{n} \leq C\sqrt{\frac{K\alpha \ln(n)}{n}}$$

Can we use UCB?

- **UCB + Empirical Distribution of Plays**

$$\mathbb{E}\left[r_\nu\left(\texttt{UCB}(\alpha), n\right)\right] \leq \frac{\mathcal{R}_\nu(\texttt{UCB}(\alpha), n)}{n} \leq C\sqrt{\frac{K\alpha \ln(n)}{n}}$$

- **UCB + Most Played Arm**

Theorem [Bubeck et al., 2009]

With the Most Play Armed as a recommendation rule, for n large enough,

$$\mathbb{E}\left[r_\nu\left(\texttt{UCB}(\alpha), n\right)\right] \leq C\sqrt{\frac{K\alpha \ln(n)}{n}}.$$

(more precise problem-dependent analysis in [Bubeck et al., 2009])

Are those results good?

$$\mathbb{E}_\nu\left[r_\nu\left(\texttt{UCB}(\alpha), n\right)\right] \simeq \sqrt{\frac{K\alpha \ln(n)}{n}}.$$

- the uniform allocation strategy can beat UCB!

Theorem [Bubeck et al., 2009]

For $n \geq 4K\ln(K)/\Delta^2_{\min}$, the simple regret decays exponentially :

$$\mathbb{E}_\nu\left[r_\nu\left(\texttt{Unif}, n\right)\right] \leq \Delta_{\max} \exp\left(-\frac{1}{8}\frac{n}{K}\Delta^2_{\min}\right)$$

- the smaller the cumulative regret, the larger the simple regret

Theorem [Bubeck et al., 2009]

$$''\mathbb{E}[r_\nu(\mathcal{A}, n)] \geq \frac{\Delta_{\min}}{2} \exp\left(-C \times \mathcal{R}_\nu(\mathcal{A}, n)\right)''$$

Can we use UCB?

Answer:

- UCB has to be coupled with an appropriate recommendation rule
- it is not guaranteed to perform better than uniform exploration...

Can we use UCB?

Answer:

- UCB has to be coupled with an appropriate recommendation rule
- it is not guaranteed to perform better than uniform exploration...

Variants of UCB?

- UCB-E for the fixed-budget setting [Audibert et al., 2010]
- LIL-UCB for the fixed-confidence setting [Jamieson et al., 2014]

Other algorithms?

- many specific algorithm for best arm identification

Fixed Budget: Sequential Halving

Input: total number of plays T

Idea: split the budget in $\log_2(K)$ phases of equal length, eliminate the worst half of the remaining arms after each phase.

Initialisation: $S_0 = \{1, \dots, K\}$;
For $r = 0$ **to** $\lceil \ln_2(K) \rceil - 1$, **do**
  sample each arm $a \in S_r$ $t_r = \left\lfloor \frac{T}{|S_r| \lceil \log_2(K) \rceil} \right\rfloor$ times;
  let $\hat{\mu}_a^r$ be the empirical mean of arm a;
  let S_{r+1} be the set of $\lceil |S_r|/2 \rceil$ arms with largest $\hat{\mu}_a^r$
Output: B_T the unique arm in $S_{\lceil \log_2(K) \rceil}$

Theorem [Karnin et al., 2013]

Letting $H_2(\nu) = \max_{a \neq a_\star} a \Delta_{[a]}^{-2}$, for any bounded bandit instance,

$$\mathbb{P}_\nu \left(B_T \neq a_\star \right) \leq 3 \log_2(K) \exp\left(- \frac{T}{8 \log_2(K) H_2(\nu)} \right).$$

Fixed Confidence: Successive Elimination

Input: risk parameter $\delta \in (0,1)$.
Idea: sample all remaining arm uniformly and perform eliminations of arms which look sub-optimal

Initialization: $\mathcal{S} = \{1, \dots, K\}$
While $|\mathcal{S}| > 1$
Draw all arms in $\mathcal{S}$. $t \leftarrow t + |\mathcal{S}|$.
$\mathcal{S} \leftarrow \mathcal{S} \backslash \{a\}$ if $\max_{i \in \mathcal{S}} \hat{\mu}_i(t) - \hat{\mu}_a(t) \geq 2\sqrt{\frac{\ln(Kt^2/\delta)}{t}}$.
Output: the unique arm $B_\tau \in \mathcal{S}$.

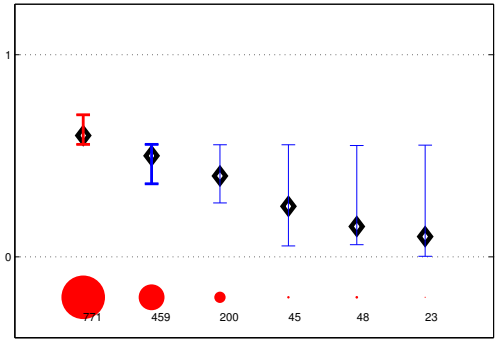
Theorem [Even-Dar et al., 2006]

Successive Elimination satisfies $\mathbb{P}_\nu(B_\tau = a_\star) \geq 1 - \delta$. Moreover,

$$\mathbb{P}_\nu\left(\tau_\delta = O\left(\sum_{a=2}^{K} \frac{1}{\Delta_a^2} \ln\left(\frac{K}{\delta \Delta_a}\right)\right)\right) \geq 1 - \delta.$$

Fixed-confidence: LUCB

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



- At round t, draw

$$B_t = \operatorname*{argmax}_b \hat{\mu}_b(t)$$

$$C_t = \operatorname*{argmax}_{c \neq B_t} \mathrm{UCB}_c(t)$$

- Stop at round t if

$$\mathrm{LCB}_{B_t}(t) > \mathrm{UCB}_{C_t}(t) - \epsilon$$

Theorem [Kalyanakrishnan et al., 2012]

For well-chosen confidence intervals, $\mathbb{P}_\nu(\mu_{B_\tau} > \mu_\star - \epsilon) \geq 1 - \delta$ and

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

Outline

Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

Minimizing the sample complexity

Context: Exponential family bandit model

$$\nu = (\nu_1, \dots, \nu_K) \quad \leftrightarrow \quad \boldsymbol{\mu} = (\mu_1, \dots, \mu_K)$$

(Bernoulli, Gaussian with known variance, Poisson...)

Algorithm: made of three components:

- ➜ sampling rule: A_t (arm to explore)
- ➜ stopping rule τ (when do we stop exploring?)
- ➜ recommendation rule: B_τ (guess for the best arm when stopping)

Objective

- a δ-correct strategy: for all $\boldsymbol{\mu}$ with a unique optimal arm,

$$\mathbb{P}_{\boldsymbol{\mu}}(B_\tau = a_\star) \geq 1 - \delta.$$

- with a small sample complexity $\mathbb{E}_{\boldsymbol{\mu}}[\tau_\delta]$.

➜ minimal sample complexity?

A first lower bound

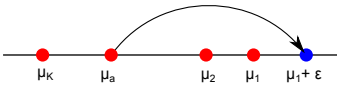
Divergence function: $\mathrm{kl}(\mu, \mu') = \mathrm{KL}(\nu_\mu, \nu_{\mu'})$.

Change of distribution lemma [Kaufmann et al., 2016]

$\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ be such that $a_\star(\boldsymbol{\mu}) \neq a_\star(\boldsymbol{\lambda})$. For any δ-correct algorithm,

$$\sum_{a=1}^{K} \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)]\mathrm{kl}(\mu_a, \lambda_a) \geq \mathrm{kl}_{\mathrm{Ber}}(\delta, 1-\delta).$$

- For any $a \in \{2, \ldots, K\}$, introducing $\boldsymbol{\lambda}$:



$$\begin{cases} \lambda_a &= \mu_1 + \epsilon \\ \lambda_i &= \mu_i, \quad \text{if} \quad i \neq a \end{cases}$$

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)]\mathrm{kl}(\mu_a, \mu_1 + \epsilon) &\geq \mathrm{kl}_{\mathrm{Ber}}(\delta, 1-\delta) \\ \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)] &\geq \frac{1}{\mathrm{kl}(\mu_a, \mu_1)} \ln\left(\frac{1}{3\delta}\right). \end{aligned}$$

A first lower bound

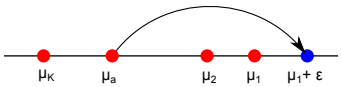
Divergence function: $\mathrm{kl}(\mu, \mu') = \frac{(\mu-\mu')^2}{2\sigma^2}$ (Gaussian distributions).

Change of distribution lemma [Kaufmann et al., 2016]

$\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ be such that $a_\star(\boldsymbol{\mu}) \neq a_\star(\boldsymbol{\lambda})$. For any δ-correct algorithm,

$$\sum_{a=1}^{K} \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)]\mathrm{kl}(\mu_a, \lambda_a) \geq \mathrm{kl}_{\mathrm{Ber}}(\delta, 1-\delta).$$

- For any $a \in \{2, \ldots, K\}$, introducing $\boldsymbol{\lambda}$:



$$\begin{cases} \lambda_a &= \mu_1 + \epsilon \\ \lambda_i &= \mu_i, \quad \text{if } i \neq a \end{cases}$$

$$\begin{aligned} \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)]\mathrm{kl}(\mu_a, \mu_1+\epsilon) &\geq \mathrm{kl}_{\mathrm{Ber}}(\delta, 1-\delta) \\ \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)] &\geq \frac{1}{\mathrm{kl}(\mu_a, \mu_1)} \ln\left(\frac{1}{3\delta}\right). \end{aligned}$$

A first lower bound

Divergence function: $\mathrm{kl}(\mu, \mu') = \mathrm{KL}(\nu_\mu, \nu_{\mu'})$.

Change of distribution lemma [Kaufmann et al., 2016]

$\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ be such that $a_\star(\boldsymbol{\mu}) \neq a_\star(\boldsymbol{\lambda})$. For any δ-correct algorithm,

$$\sum_{a=1}^{K} \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)]\mathrm{kl}(\mu_a, \lambda_a) \geq \mathrm{kl}_{\mathrm{Ber}}(\delta, 1-\delta).$$

- One obtains

Theorem [Kaufmann et al., 2016]

For any δ-correct algorithm,

$$\mathbb{E}_{\boldsymbol{\mu}}[\tau] \geq \left(\frac{1}{\mathrm{kl}(\mu_1, \mu_2)} + \sum_{a=2}^{K} \frac{1}{\mathrm{kl}(\mu_a, \mu_1)} \right) \ln\left(\frac{1}{3\delta}\right).$$

A first lower bound

Divergence function: $\mathrm{kl}(\mu, \mu') = \mathrm{KL}(\nu_\mu, \nu_{\mu'})$.

Change of distribution lemma [Kaufmann et al., 2016]

$\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ be such that $a_\star(\boldsymbol{\mu}) \neq a_\star(\boldsymbol{\lambda})$. For any δ-correct algorithm,

$$\sum_{a=1}^{K} \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)]\mathrm{kl}(\mu_a, \lambda_a) \geq \mathrm{kl}_{\mathrm{Ber}}(\delta, 1-\delta).$$

- One obtains

Theorem [Kaufmann et al., 2016]

For any δ-correct algorithm,

$$\mathbb{E}_{\boldsymbol{\mu}}[\tau] \geq \left(\frac{1}{\mathrm{kl}(\mu_1, \mu_2)} + \sum_{a=2}^{K} \frac{1}{\mathrm{kl}(\mu_a, \mu_1)} \right) \ln\left(\frac{1}{3\delta}\right).$$

➜ not tight enough...

The best possible lower bound

Change of distribution lemma [Kaufmann et al., 2016]

$\boldsymbol{\mu}$ and $\boldsymbol{\lambda}$ be such that $a_\star(\boldsymbol{\mu}) \neq a_\star(\boldsymbol{\lambda})$. For any δ-correct algorithm,

$$\sum_{a=1}^{K} \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)]\mathrm{kl}(\mu_a, \lambda_a) \geq \mathrm{kl}_{\mathsf{Ber}}(\delta, 1-\delta).$$

- Let $\mathrm{Alt}(\boldsymbol{\mu}) = \{\boldsymbol{\lambda} : a_\star(\boldsymbol{\lambda}) \neq a_\star(\boldsymbol{\mu})\}$.

$$\begin{aligned}
\inf_{\boldsymbol{\lambda}\in\mathrm{Alt}(\boldsymbol{\mu})} \sum_{a=1}^{K} \mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)]\mathrm{kl}(\mu_a, \lambda_a) &\geq \mathrm{kl}_{\mathsf{Ber}}(\delta, 1-\delta) \\
\mathbb{E}_{\boldsymbol{\mu}}[\tau] \times \inf_{\boldsymbol{\lambda}\in\mathrm{Alt}(\boldsymbol{\mu})} \sum_{a=1}^{K} \frac{\mathbb{E}_{\boldsymbol{\mu}}[N_a(\tau)]}{\mathbb{E}_{\boldsymbol{\mu}}[\tau]}\mathrm{kl}(\mu_a, \lambda_a) &\geq \ln\left(\frac{1}{3\delta}\right) \\
\mathbb{E}_{\boldsymbol{\mu}}[\tau] \times \left(\sup_{w\in\Sigma_K} \inf_{\boldsymbol{\lambda}\in\mathrm{Alt}(\boldsymbol{\mu})} \sum_{a=1}^{K} w_a \mathrm{kl}(\mu_a, \lambda_a)\right) &\geq \ln\left(\frac{1}{3\delta}\right)
\end{aligned}$$

The best possible lower bound

Theorem [Garivier and Kaufmann, 2016]

For any δ-PAC algorithm,

$$\mathbb{E}_{\boldsymbol{\mu}}[\tau] \geq T_\star(\boldsymbol{\mu}) \ln\left(\frac{1}{3\delta}\right),$$

where

$$T_\star(\boldsymbol{\mu})^{-1} = \sup_{w \in \Sigma_K} \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\boldsymbol{\mu})} \left(\sum_{a=1}^{K} w_a \mathrm{kl}(\mu_a, \lambda_a) \right).$$

Moreover, the vector of optimal proportions,

$$w_\star(\boldsymbol{\mu}) = \operatorname*{argmax}_{w \in \Sigma_K} \inf_{\lambda \in \mathrm{Alt}(\boldsymbol{\mu})} \left(\sum_{a=1}^{K} w_a \mathrm{kl}(\mu_a, \lambda_a) \right)$$

is well-defined, and can be computed efficiently.

Outline

Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

How to match the lower bound? Sampling rule.

$\hat{\boldsymbol{\mu}}(t) = (\hat{\mu}_1(t), \dots, \hat{\mu}_K(t))$: vector of empirical means

- Introducing

$$U_t = \left\{ a : N_a(t) < \sqrt{t} \right\},$$

one has

$$A_{t+1} \in \begin{cases} \underset{a \in U_t}{\operatorname{argmin}} \; N_a(t) \text{ if } U_t \neq \emptyset & (\textit{forced exploration}) \\ \underset{1 \leq a \leq K}{\operatorname{argmax}} \left[(w_\star(\hat{\boldsymbol{\mu}}(t)))_a - \frac{N_a(t)}{t} \right] & (\textit{tracking}) \end{cases}$$

Lemma

Under the Tracking sampling rule,

$$\mathbb{P}_{\boldsymbol{\mu}} \left(\lim_{t \to \infty} \frac{N_a(t)}{t} = (w_\star(\boldsymbol{\mu}))_a \right) = 1.$$

How to match the lower bound? Stopping rule.

- a Generalized Likelihood Ratio:

$$\hat{Z}(t) = \ln \frac{\ell\left(X_1, \dots, X_t; \hat{\boldsymbol{\mu}}(t)\right)}{\max\limits_{\boldsymbol{\lambda} \in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))} \ell(X_1, \dots, X_t; \boldsymbol{\lambda})} = \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))} \sum_{a=1}^{K} N_a(t) \mathrm{kl}(\hat{\mu}_a(t), \lambda_a)$$

➜ high value of $\hat{Z}(t)$ rejects the hypothesis "$\boldsymbol{\mu} \in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))$".

Stopping and recommendation rule

$$\begin{aligned} \tau_\delta &= \inf\left\{ t \in \mathbb{N} : \hat{Z}(t) > \beta(t, \delta) \right\} \\ B_\tau &= \operatorname*{argmax}_{a=1,\dots,K} \hat{\mu}_a(\tau). \end{aligned}$$

(can be traced back to [Chernoff, 1959])

How to match the lower bound? Stopping rule.

- a Generalized Likelihood Ratio:

$$\hat{Z}(t) = \ln \frac{\ell\left(X_1, \dots, X_t; \hat{\boldsymbol{\mu}}(t)\right)}{\max_{\boldsymbol{\lambda} \in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))} \ell(X_1, \dots, X_t; \boldsymbol{\lambda})} = \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))} \sum_{a=1}^{K} N_a(t) \mathrm{kl}(\hat{\mu}_a(t), \lambda_a)$$

➜ high value of $\hat{Z}(t)$ rejects the hypothesis "$\boldsymbol{\mu} \in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))$".

Stopping and recommendation rule

$$\begin{aligned} \tau_\delta &= \inf\left\{ t \in \mathbb{N} : \hat{Z}(t) > \beta(t, \delta) \right\} \\ B_\tau &= \operatorname*{argmax}_{a=1,\dots,K} \hat{\mu}_a(\tau). \end{aligned}$$

(can be traced back to [Chernoff, 1959])

➜ How to pick the threshold $\beta(t, \delta)$?

A δ-correct stopping rule

$$\hat{Z}(t) = \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))} \sum_{a=1}^{K} N_a(t) \mathrm{kl}(\hat{\mu}_a(t), \lambda_a)$$

$$= \min_{b \neq B_t} \inf_{\{\boldsymbol{\lambda}: \lambda_{B_t} \leq \lambda_b\}} \sum_{a=1}^{K} N_a(t) \mathrm{kl}(\hat{\mu}_a(t), \lambda_a)$$

bla

A δ-correct stopping rule

$$\begin{aligned}\hat{Z}(t) &= \inf_{\boldsymbol{\lambda}\in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))} \sum_{a=1}^{K} N_a(t)\mathrm{kl}(\hat{\mu}_a(t),\lambda_a) \\ &= \min_{b\neq B_t} \inf_{\{\boldsymbol{\lambda}:\lambda_{B_t}\leq\lambda_b\}} \left[N_{B_t}(t)\mathrm{kl}(\hat{\mu}_{B_t}(t),\lambda_{B_t}) + N_b(t)\mathrm{kl}(\hat{\mu}_b(t),\lambda_b) \right]\end{aligned}$$

A δ-correct stopping rule

$$\hat{Z}(t) = \inf_{\boldsymbol{\lambda} \in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))} \sum_{a=1}^{K} N_a(t) \mathrm{kl}(\hat{\mu}_a(t), \lambda_a)$$

$$= \min_{b \neq B_t} \inf_{\{\boldsymbol{\lambda} : \lambda_{B_t} \leq \lambda_b\}} \left[N_{B_t}(t) \mathrm{kl}(\hat{\mu}_{B_t}(t), \lambda_{B_t}) + N_b(t) \mathrm{kl}(\hat{\mu}_b(t), \lambda_b) \right]$$

$$\mathbb{P}\left(B_{\tau_\delta} \neq a_\star\right) \leq \mathbb{P}\left(\exists t \in \mathbb{N}_\star, \exists a \neq a_\star : B_t = a, \hat{Z}(t) > \beta(t,\delta)\right)$$

$$\leq \mathbb{P}\left(\exists t \in \mathbb{N}_\star, \exists a \neq a_\star : \inf_{\lambda_a \leq \lambda_{a_\star}} \sum_{i \in \{a, a_\star\}} N_i(t) \mathrm{kl}(\hat{\mu}_i(t), \lambda_i) > \beta(t,\delta)\right)$$

$$\leq \sum_{a \neq a_\star} \underbrace{\mathbb{P}\left(\exists t \in \mathbb{N}_\star : N_a(t) \mathrm{kl}(\hat{\mu}_a(t), \mu_a) + N_{a_\star}(t) \mathrm{kl}(\hat{\mu}_{a_\star}(t), \mu_{a_\star}) > \beta(t,\delta)\right)}_{\text{requires simultaneous deviations on the two arms}}$$

Bernoulli case: [Garivier and Kaufmann, 2016]

Exponential families: [Kaufmann and Koolen, 2018]

An asymptotically optimal algorithm

Theorem

The Track-and-Stop strategy, that uses

- the Tracking sampling rule
- the GLRT stopping rule with

$$\beta(t,\delta) \simeq \ln\left(\frac{K-1}{\delta}\right) + 2\ln\ln\left(\frac{K-1}{\delta}\right) + 6\ln(\ln(t))$$

- and recommends $B_\tau = \underset{a=1\dots K}{\operatorname{argmax}}\ \hat{\mu}_a(\tau)$

is δ-PAC for every $\delta \in]0,1[$ and satisfies

$$\limsup_{\delta\to 0} \frac{\mathbb{E}_\mu[\tau_\delta]}{\ln(1/\delta)} = T_\star(\mu).$$

An asymptotically optimal algorithm

Theorem

The Track-and-Stop strategy, that uses

- the Tracking sampling rule
- the GLRT stopping rule with

$$\beta(t,\delta) \simeq \ln\left(\frac{K-1}{\delta}\right) + 2\ln\ln\left(\frac{K-1}{\delta}\right) + 6\ln(\ln(t))$$

- and recommends $B_\tau = \underset{a=1\dots K}{\operatorname{argmax}}\ \hat{\mu}_a(\tau)$

is δ-PAC for every $\delta \in]0,1[$ and satisfies

$$\limsup_{\delta\to 0} \frac{\mathbb{E}_\mu[\tau_\delta]}{\ln(1/\delta)} = T_\star(\mu).$$

Why?

$$\tau_\delta = \inf\left\{ t \in \mathbb{N}_\star : \inf_{\lambda\in \mathrm{Alt}(\hat{\mu}(t))} \sum_{a=1}^{K} N_a(t)\mathrm{kl}\left(\hat{\mu}_a(t),\lambda_a\right) > \beta(t,\delta)\right\}$$

An asymptotically optimal algorithm

Theorem

The Track-and-Stop strategy, that uses

- the Tracking sampling rule
- the GLRT stopping rule with

$$\beta(t,\delta) \simeq \ln\left(\frac{K-1}{\delta}\right) + 2\ln\ln\left(\frac{K-1}{\delta}\right) + 6\ln(\ln(t))$$

- and recommends $B_\tau = \underset{a=1\dots K}{\operatorname{argmax}}\ \hat{\mu}_a(\tau)$

is δ-PAC for every $\delta \in]0,1[$ and satisfies

$$\limsup_{\delta\to 0} \frac{\mathbb{E}_{\boldsymbol{\mu}}[\tau_\delta]}{\ln(1/\delta)} = T_\star(\boldsymbol{\mu}).$$

Why?

$$\tau_\delta = \inf\left\{ t \in \mathbb{N}_\star : t \times \inf_{\boldsymbol{\lambda}\in \mathrm{Alt}(\hat{\boldsymbol{\mu}}(t))} \sum_{a=1}^{K} \frac{N_a(t)}{t} \mathrm{kl}\left(\hat{\mu}_a(t), \lambda_a\right) > \beta(t,\delta) \right\}$$

An asymptotically optimal algorithm

Theorem

The Track-and-Stop strategy, that uses

- the Tracking sampling rule
- the GLRT stopping rule with

$$\beta(t,\delta) \simeq \ln\left(\frac{K-1}{\delta}\right) + 2\ln\ln\left(\frac{K-1}{\delta}\right) + 6\ln(\ln(t))$$

- and recommends $B_\tau = \underset{a=1\dots K}{\operatorname{argmax}}\ \hat{\mu}_a(\tau)$

is δ-PAC for every $\delta \in]0,1[$ and satisfies

$$\limsup_{\delta\to 0} \frac{\mathbb{E}_{\boldsymbol{\mu}}[\tau_\delta]}{\ln(1/\delta)} = T_\star(\boldsymbol{\mu}).$$

Why?

$$\tau_\delta \simeq \inf\left\{ t \in \mathbb{N}_\star : t \times \inf_{\boldsymbol{\lambda}\in \mathrm{Alt}(\boldsymbol{\mu})} \sum_{a=1}^{K} (w_\star(\boldsymbol{\mu}))_a \mathrm{kl}\left(\mu_a, \lambda_a\right) > \beta(t,\delta) \right\}$$

An asymptotically optimal algorithm

Theorem

The Track-and-Stop strategy, that uses

- the Tracking sampling rule
- the GLRT stopping rule with

$$\beta(t,\delta) \simeq \ln\left(\frac{K-1}{\delta}\right) + 2\ln\ln\left(\frac{K-1}{\delta}\right) + 6\ln(\ln(t))$$

- and recommends $B_\tau = \underset{a=1\dots K}{\operatorname{argmax}}\ \hat{\mu}_a(\tau)$

is δ-PAC for every $\delta \in]0,1[$ and satisfies

$$\limsup_{\delta\to 0} \frac{\mathbb{E}_{\boldsymbol{\mu}}[\tau_\delta]}{\ln(1/\delta)} = T_\star(\boldsymbol{\mu}).$$

Why?

$$\tau_\delta \simeq \inf\left\{ t \in \mathbb{N}_\star : t \times T_\star^{-1}(\boldsymbol{\mu}) > \beta(t,\delta) \right\}$$

Numerical experiments

- $\boldsymbol{\mu}_1 = [0.5\ 0.45\ 0.43\ 0.4]$, such that

$$w_\star(\boldsymbol{\mu}_1) = [0.417\ 0.390\ 0.136\ 0.057]$$

- $\boldsymbol{\mu}_2 = [0.3\ 0.21\ 0.2\ 0.19\ 0.18]$, such that

$$w_\star(\boldsymbol{\mu}_2) = [0.336\ 0.251\ 0.177\ 0.132\ 0.104]$$

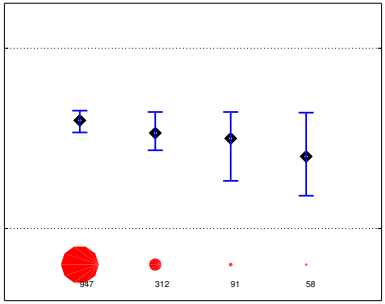
In practice, set the threshold to $\beta(t,\delta) = \ln\left(\frac{\ln(t)+1}{\delta}\right)$.

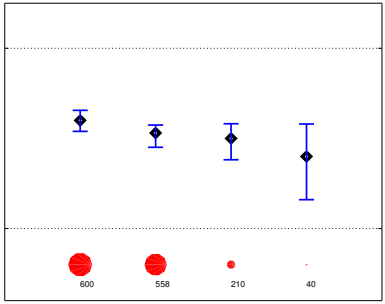
	Track-and-Stop	GLRT-SE	KL-LUCB	KL-SE
$\boldsymbol{\mu}_1$	4052	4516	8437	9590
$\boldsymbol{\mu}_2$	1406	3078	2716	3334

Table: Expected number of draws $\mathbb{E}_{\boldsymbol{\mu}}[\tau_\delta]$ for $\delta = 0.1$, averaged over $N = 3000$ experiments.

Observations and improvements

- TaS: a lower-bound inspired algorithm
- KL-UCB versus TaS: very different sampling rules!





- Two recent improvements of the Tracking sampling rule:
  - ➜ relax the need for computing the optimal weights at every round [Ménard, 2019]
  - ➜ get rid of the forced exploration by using Upper Confidence Bounds [Degenne et al., 2019]

Open problems

- Unlike previously mentioned strategies, the sampling rule of TaS is **anytime**, i.e. does not depend on a budget T or a risk parameter δ.
  - ➜ is it also a good strategy in the fixed budget setting?
  - ➜ can control $\mathbb{E}[r_\nu(\mathrm{TaS}, t)]$ for any t?

- Fixed-budget setting: no exactly matching upper and lower bound
  best lower bounds: [Carpentier and Locatelli, 2016]

- Top-Two Thompson Sampling [Russo, 2016]: a Bayesian (anytime) strategy that is optimal in a different (Bayesian, asymptotic) sense
  - ➜ can we obtain frequentist guarantees for this algorithm?

Beyond Best Arm Identification

Outline

Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

Active Identification in Bandit Models

Context: exponential family bandit model

$$\nu \leftrightarrow \boldsymbol{\mu} = (\mu_1, \dots, \mu_K) \in \mathcal{I}^K$$

Goal: Given M regions of $\mathcal{I}^K$, $\mathcal{R}_1, \dots, \mathcal{R}_M$, the goal is to identify one region to which $\boldsymbol{\mu}$ belongs.

Formalization: build a

- ➜ sampling rule (A_t)
- ➜ stopping rule τ
- ➜ recommendation rule $\hat{\imath}_\tau \in \{1, \dots, M\}$

such that, for some risk parameter δ,

$$\mathbb{P}_{\boldsymbol{\mu}}\left(\boldsymbol{\mu} \notin \mathcal{R}_{\hat{\imath}_\tau}\right) \leq \delta \quad \text{and} \quad \mathbb{E}_{\boldsymbol{\mu}}[\tau] \text{ is small.}$$

Active Identification in Bandit Models

Context: exponential family bandit model

$$\nu \leftrightarrow \boldsymbol{\mu} = (\mu_1, \dots, \mu_K) \in \mathcal{I}^K$$

Goal: Given M regions of $\mathcal{I}^K$, $\mathcal{R}_1, \dots, \mathcal{R}_M$, the goal is to identify one region to which $\boldsymbol{\mu}$ belongs.

Two cases:

- $\mathcal{R}_1, \dots, \mathcal{R}_M$ form a partition : a lower-bound inspired sampling rule + a GLRT stopping rule essentially works
  ⚠computing the optimal allocation may be difficult
  [Juneja and Krishnasamy, 2019, Kaufmann and Koolen, 2018]
- $\mathcal{R}_1, \dots, \mathcal{R}_M$ are overlapping regions : a Track-and-Stop approach does not always work [Degenne and Koolen, 2019]

Outline

Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

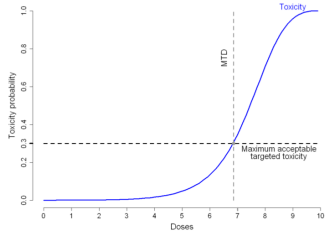
Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

Bandits and thresholds

Anomaly detection: given a threshold θ:

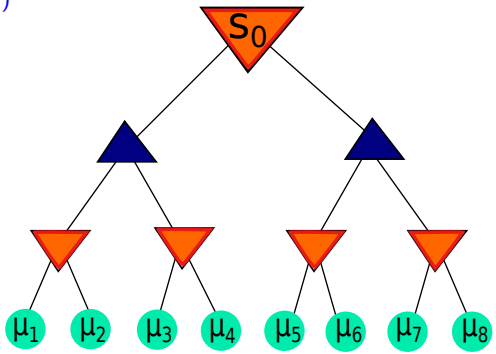
- find all the arms whose mean is below θ [Locatelli et al., 2016]
- find whether there is an arm with mean below θ [Kaufmann et al., 2018]

Phase I clinical trial: find the arm with mean closest to the threshold... with increasing means. [Garivier et al., 2017]



$$\mathcal{R}_a = \left\{ \boldsymbol{\lambda} \in \texttt{Inc} : a = \operatorname*{argmin}_i |\theta - \lambda_i| \right\}$$

Find the best move at the root of a game tree by actively sampling its leaves. $s_\star = \underset{s \in \mathcal{C}(s_0)}{\text{argmax}}\ V_s$.



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

➜ more details later

Bandit for Optimization in a Larger Space

Outline

Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

Bandit problems from an optimization perspective

$$f : \{1, \dots, K\} \longrightarrow \mathbb{R} \qquad \max_{a=1,\dots,K} f(a) \ ?$$

Sequential evaluations: at time t, choose $A_t \in \{1, \dots, K\}$, observe

$$X_t \sim \nu_{A_t} \quad \text{where} \quad \nu_a \quad \text{has mean} \quad f(a).$$

After T observations,

Minimize the cumulative regret

$$\text{minimize} \ \ \mathbb{E}\left[\sum_{t=1}^{T}(f(a_\star) - f(A_t))\right]$$

Minimize the simple regret (optimization error)

If B_T is a guess of the argmax

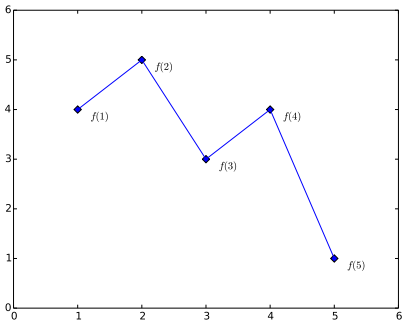
$$\text{minimize} \ \ \mathbb{E}\left[f(a_\star) - f(B_T)\right]$$

Bandit problems from an optimization perspective

$$f : \{1, \dots, K\} \longrightarrow \mathbb{R} \qquad \max_{a=1,\dots,K} f(a) \ ?$$

Sequential evaluations: at time t, choose $A_t \in \{1, \dots, K\}$, observe

$$X_t \sim \nu_{A_t} \quad \text{where} \quad \nu_a \quad \text{has mean} \quad f(a).$$



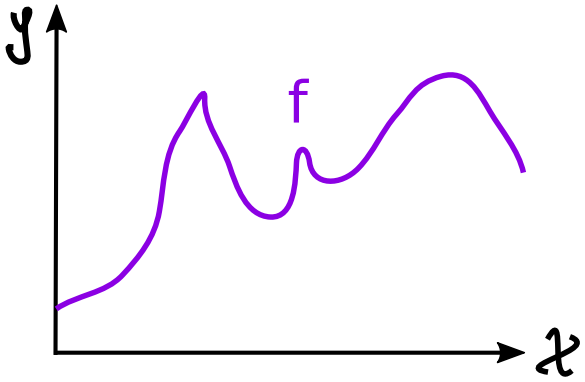
sequential optimization of a discrete function based on noisy observations

General Sequential (Noisy) Optimization

$$f : \mathcal{X} \longrightarrow \mathbb{R} \qquad \max_{x \in \mathcal{X}} f(x) \ ?$$

Sequential evaluations: at time t, choose $x_t \in \mathcal{X}$, observe

$$y_t = f(x_t) + \epsilon_t$$



sequential optimization of a black-box function based on noisy observations

General Sequential (Noisy) Optimization

$$f : \mathcal{X} \longrightarrow \mathbb{R} \qquad \max_{x \in \mathcal{X}} f(x) \ ?$$

Sequential evaluations: at time t, choose $x_t \in \mathcal{X}$, observe

$$y_t = f(x_t) + \epsilon_t$$

After T observations,

Minimize the cumulative regret

$$\text{minimize} \quad \mathbb{E}\left[\sum_{t=1}^{T} (f(x_\star) - f(x_t))\right]$$

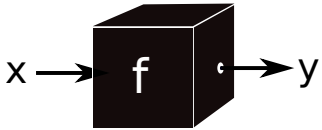
Minimize the simple regret (optimization error)

If z_T is a guess of the argmax

$$\text{minimize} \quad \mathbb{E}\left[f(x_\star) - f(z_T)\right]$$

Black Box Optimization

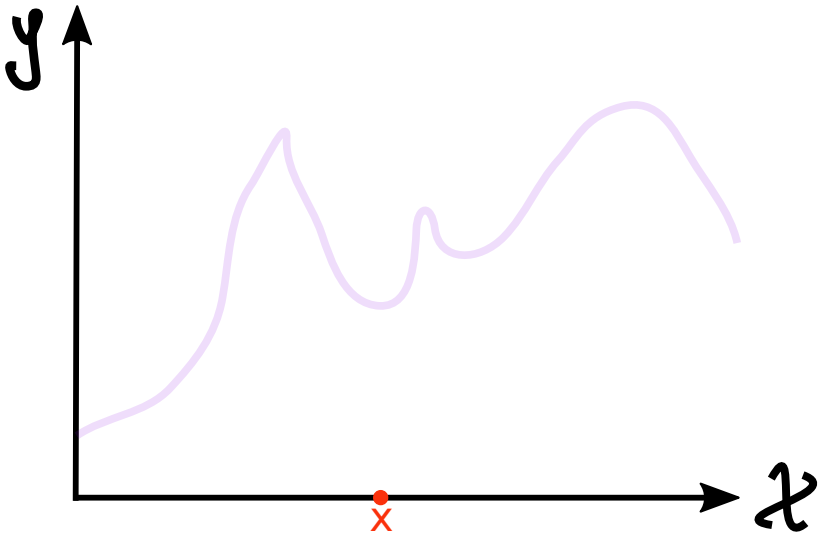
- learning based on (costly, noisy) function evaluations only!
- no access to gradients of f



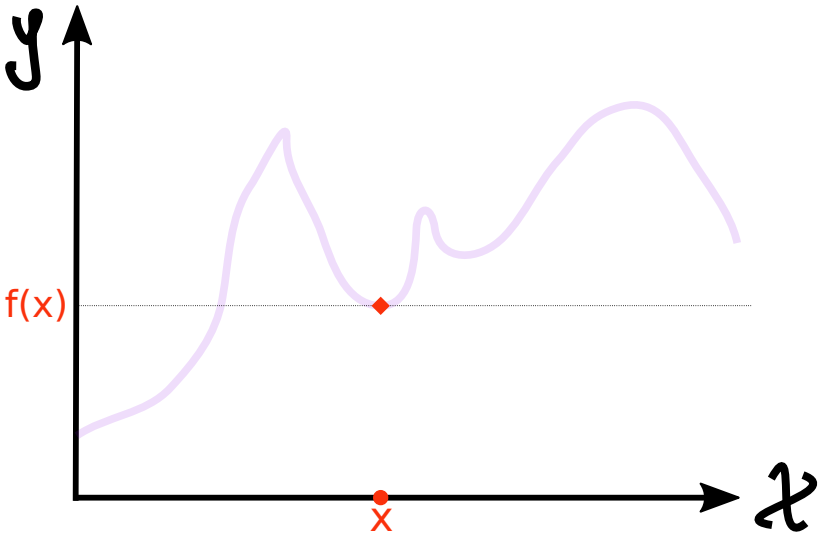
Examples of function f

- a costly PDE solver (numerical experiments)
- a simulator of the effect of a chemical compound (drug discovery)
- training and evaluation a neural network (hyper-parameter optimization)

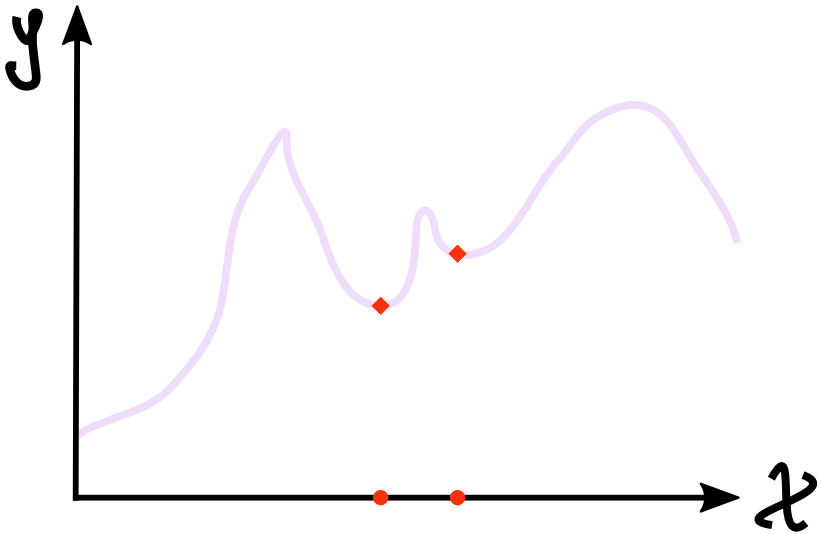
What we want to do



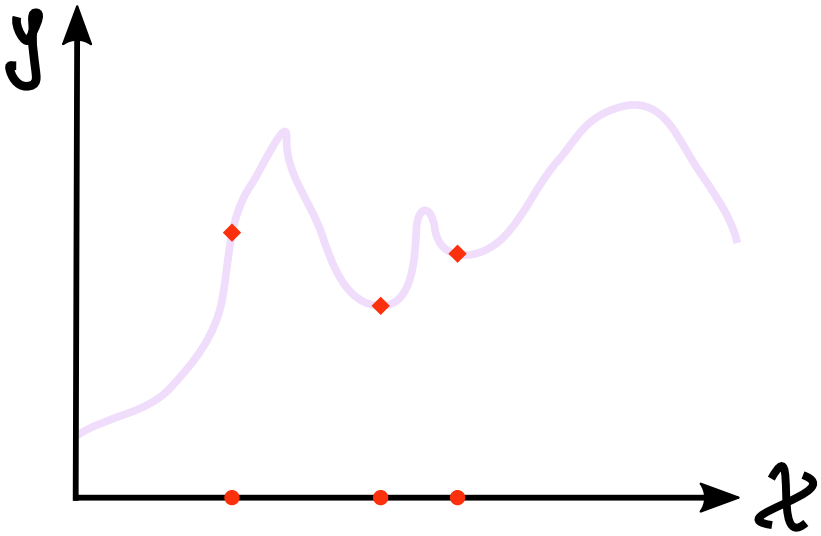
What we want to do



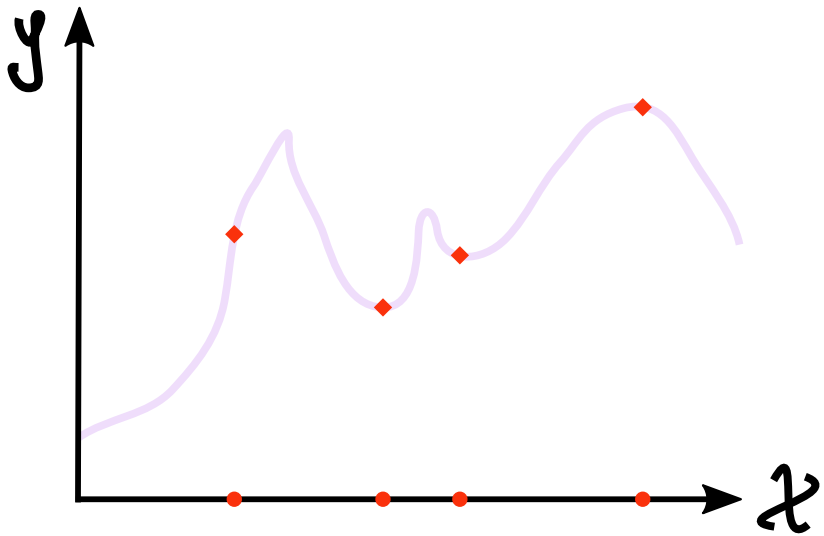
What we want to do



What we want to do

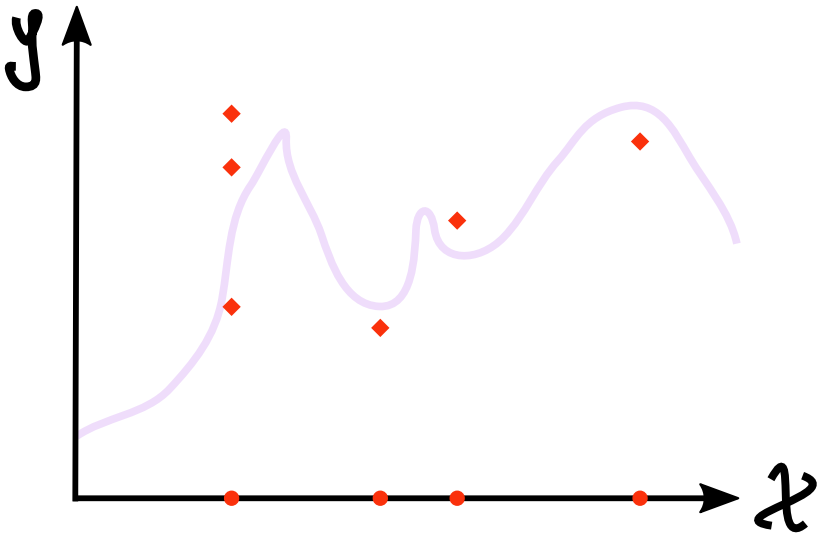


What we want to do



How to choose the next querry?

What we want to do



How to choose the next querry?

Outline

Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

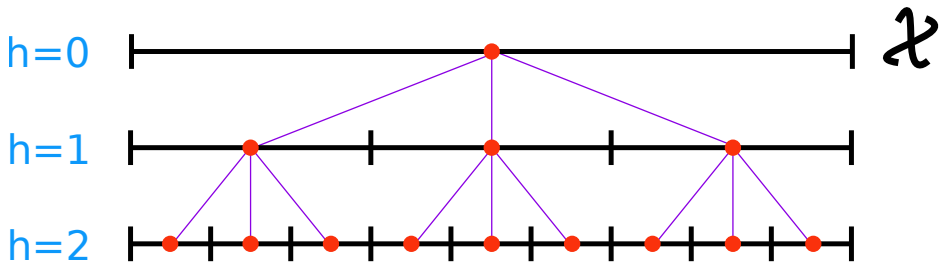
Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

Hierarchical partitioning

Bandit in metric spaces [Kleinberg et al., 2008]
$\mathcal{X}$-armed bandits [Bubeck et al., 2011]

Idea: Partition the space, and adaptatively choose in which cell to sample



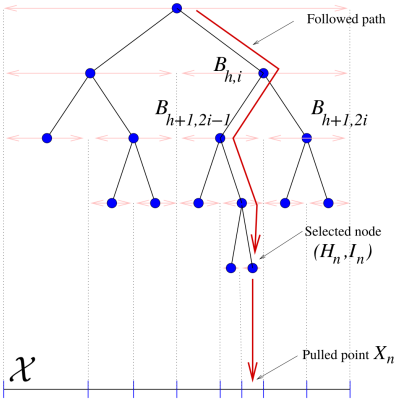
- For any **depth** h, $\mathcal{X}$ is partitioned in K^h cells $(\mathcal{P}_{h,i})_{0\leq K^h-1}$.
- K-ary tree $\mathcal{T}$ where depth $h=0$ is the whole $\mathcal{X}$.

Hierarchical Optimistic Optimization (HOO)

Assumptions (given a metric $\ell(x, y)$)

- $f(x_\star) - f(y) \leq f(x_\star) - f(x) + \max\{f(x_\star) - f(x); \ell(x, y)\}$
- $\sup_{(x,y)\in\mathcal{P}_{h,i}} \ell(x, y) \leq \nu\rho^h$.

Idea: Use Upper-Confidence Bounds on the maximum values of the function in each cell to guide exploration



$$U_{(h,i)}(t) = \hat{\mu}_{(h,i)}(t) + \sqrt{\frac{2\ln(t)}{N_{(h,i)}(t)}} + \nu\rho^h$$

$$B_{(h,i)}(t) = \min\Big\{U_{(h,i)}(t); B_{(h+1,2i)}(t); B_{(h+1,2i+1)}(t)\Big\}$$

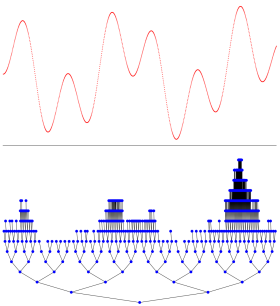
[Bubeck et al., 2011]

Results

Cumulative regret of HOO

$$\mathbb{E}\left[\sum_{t=1}^{T}(f(x_\star)-f(x_t))\right] \leq CT^{\frac{d+1}{d+2}}(\ln(T))^{\frac{1}{d+2}}$$

for some near-optimality dimension d.



➜ how to turn this into an optimizer?

$z_T \sim \mathcal{U}(x_1,\dots,x_T)$, then

$$\mathbb{E}[f(x_\star)-f(z_T)] = \frac{\mathcal{R}(\texttt{HOO},T)}{T} \leq C\left(\frac{\ln(T)}{T}\right)^{\frac{1}{d+2}}$$

a tree built by HOO

- **Many variants!**

DOO, SOO [Munos, 2011], StoSOO [Valko et al., 2013], POO [Grill et al., 2015], GPO [Shang et al., 2019]...

Outline

Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

Gaussian Process Regression

Assumption. the function f is drawn from some Gaussian Process :

$$f \sim \mathcal{GP}(0, k(x, y)).$$

i.e. for any distinct points $x_1, \dots, x_\ell$ in $\mathcal{X}$,

$$\begin{pmatrix} f(x_1) \\ f(x_2) \\ \dots \\ f(x_\ell) \end{pmatrix} \sim \mathcal{N}(0, K) \quad \text{where} \quad K = (k(x_i, x_j))_{1 \leq i,j \leq \ell}$$

Bayesian inference

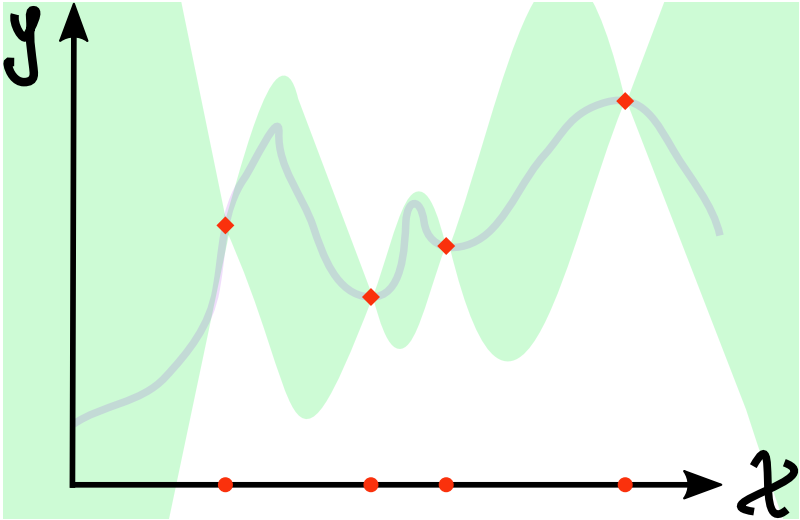
Given some (possibly noisy) observations of f in $x_1, \dots, x_t$, the posterior on all the function value in any point is Gaussian

$$f(y)|x_1, \dots, x_t \sim \mathcal{N}\left(\mu_t(y), \sigma_t(y)^2\right)$$

[Rasmussen and Williams, 2005]

Bayesian Optimization

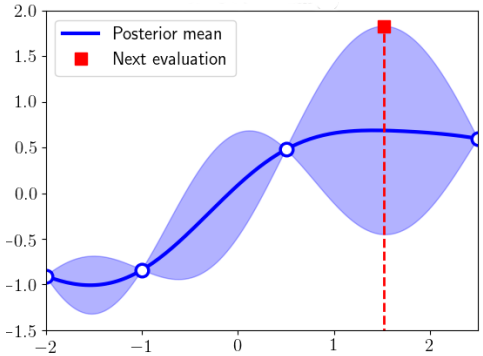
➜ use the current GP posterior to pick the new point to select



Example: $[\mu_t(y) \pm \beta\sigma_t(y)]$ is a kind of confidence interval on $f(y)$.

GP-UCB [Srinivas et al., 2012] selects at round $t+1$

$$x_{t+1} = \underset{x \in \mathcal{X}}{\operatorname{argmax}} \ \mu_t(x) + \sqrt{\beta(t,\delta)}\sigma_t(x)$$



➜ Bayesian and frequentist guarantees in terms of cumulative regret for different $\beta(t,\delta)$: $\mathbb{P}\left(\mathcal{R}_T(\texttt{GP-UCB},T) \leq C\sqrt{T\beta(T,\delta)\gamma_T}\right) \geq 1-\delta$.

BO Algorithms

More generally, many Bayesian Optimization algorithms optimize an acquisition function that depends on the posterior and select

$$x_{t+1} = \operatorname*{argmax}_{x \in \mathcal{X}} \alpha_t(x)$$

Many other acquisition functions: [Shahriari et al., 2016]

- Expected improvement
- Probability of improvement
- Entropy Search ...

Remark: optimization the acquisition function is another (non-trivial) optimization problem!

- Thompson Sampling?

Bandit Tools for Planning in Games

Outline

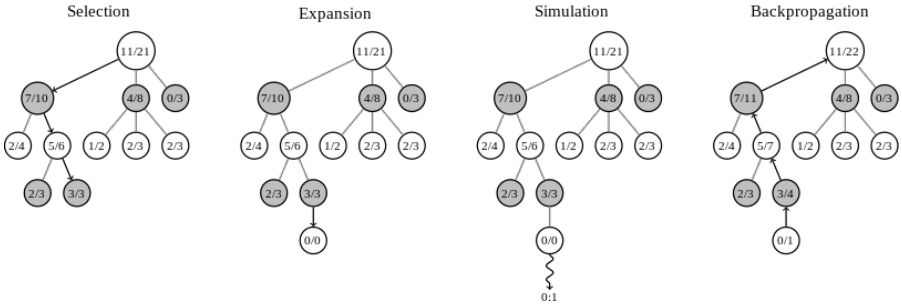
Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

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: [Kocsis and Szepesvári, 2006]

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

The UCT algorithm

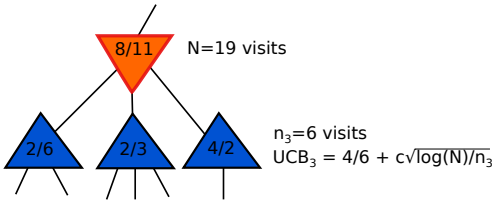
$N(s)$: number of visits of node s

$S(s)$: number of visits finishing ending with the root player winning

UCT in a MaxMin Tree

In a MAX node s ($=$ root player move), go towards the children

$$\underset{c \in \mathcal{C}(s)}{\operatorname{argmax}} \ \frac{S(c)}{N(c)} + c\sqrt{\frac{\ln(N(s))}{N(c)}}$$



The UCT algorithm

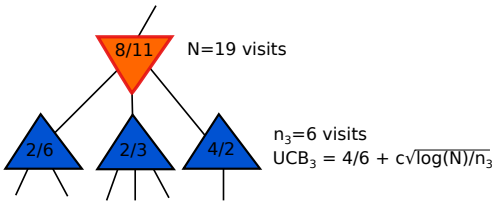
$N(s)$: number of visits of node s

$S(s)$: number of visits finishing ending with the root player winning

UCT in a MaxMin Tree

In a MIN node s ($=$ adversary move), go towards the children

$$\underset{c \in \mathcal{C}(s)}{\operatorname{argmin}} \frac{S(c)}{N(c)} - c\sqrt{\frac{\ln(N(s))}{N(c)}}$$



The UCT algorithm

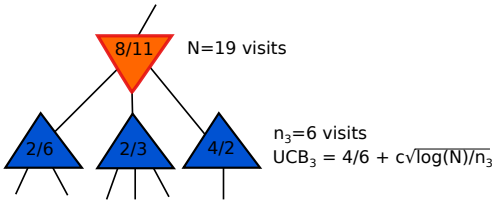
$N(s)$: number of visits of node s

$S(s)$: number of visits finishing ending with the root player winning

UCT in a MaxMin Tree

In a MAX node s ($=$ root player move), go towards the children

$$\underset{c \in \mathcal{C}(s)}{\operatorname{argmax}} \ \frac{S(c)}{N(c)} + c\sqrt{\frac{\ln(N(s))}{N(c)}}$$



The UCT algorithm

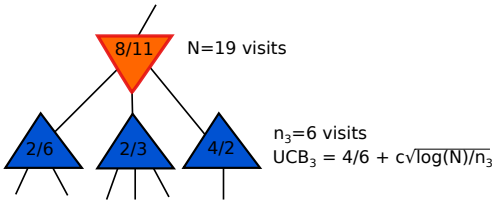
$N(s)$: number of visits of node s

$S(s)$: number of visits finishing ending with the root player winning

UCT in a MaxMin Tree

In a MAX node s (= root player move), go towards the children

$$\underset{c \in \mathcal{C}(s)}{\operatorname{argmax}} \ \frac{S(c)}{N(c)} + c\sqrt{\frac{\ln(N(s))}{N(c)}}$$



+ first Go AI based on variants of UCT (+ heuristics)

The UCT algorithm

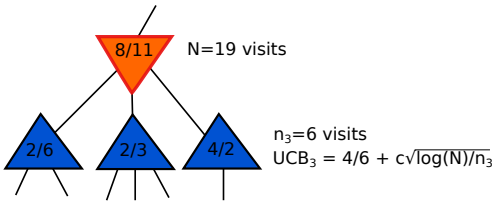
$N(s)$: number of visits of node s

$S(s)$: number of visits finishing ending with the root player winning

UCT in a MaxMin Tree

In a MAX node s ($=$ root player move), go towards the children

$$\underset{c \in \mathcal{C}(s)}{\operatorname{argmax}} \ \frac{S(c)}{N(c)} + c\sqrt{\frac{\ln(N(s))}{N(c)}}$$



- UCT is *not* based on statistically-valid confidence intervals
- no sample complexity guarantees
- should we really minimize rewards?

Outline

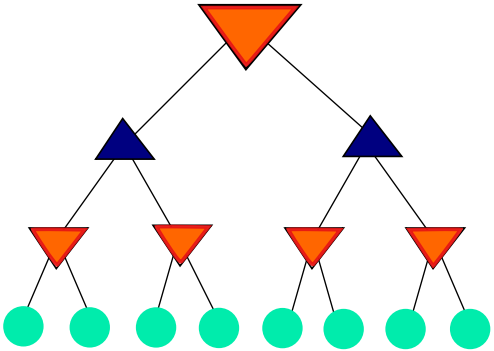
Finding the Best Arm in a Bandit Model
- Algorithms for Best Arm Identification
- Performance Lower Bounds
- An asymptotically optimal algorithm

Beyond Best Arm Identification
- Active Identification in Bandit Models
- Examples

Bandit for Optimization in a Larger Space
- Black-Box Optimization
- Hierarchical Bandits
- Bayesian Optimization

Bandit Tools for Planning in Games
- Upper Confidence Bounds for Trees
- BAI tools for Planning in Games

A simple model for MCTS



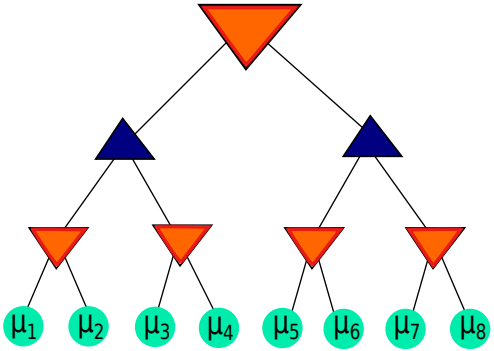
A fixed MAXMIN game tree $\mathcal{T}$, with leaves $\mathcal{L}$.

MAX node (= root player move)

MIN node (= adversary move)

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

A simple model for MCTS

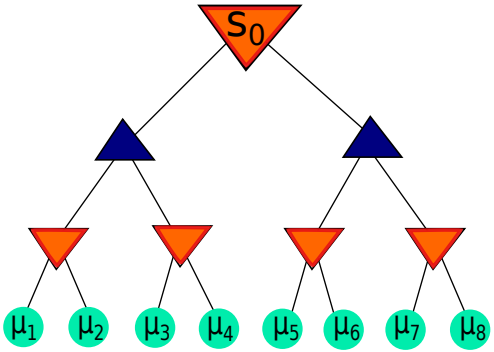


At round t a **MCTS algorithm**:

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

Goal



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

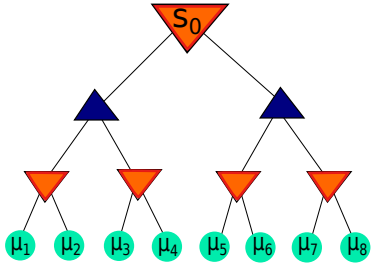
$$V_s = \begin{cases} \mu_s & \text{if s } \in \mathcal{L}, \\ \max_{c \in \mathcal{C}(s)} V_c & \text{if s is a MAX node,} \\ \min_{c \in \mathcal{C}(s)} V_c & \text{if s is a MIN node.} \end{cases}$$

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

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:

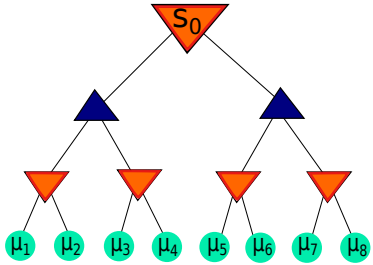
$$\mathbb{P}(\boldsymbol{V}_{\hat{s}_\tau} \geq \boldsymbol{V}_{s^*} - \boldsymbol{\epsilon}) \geq \mathbf{1} - \boldsymbol{\delta}$$

with a small sample complexity $\boldsymbol{\tau}$. [Teraoka et al., 2014]

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

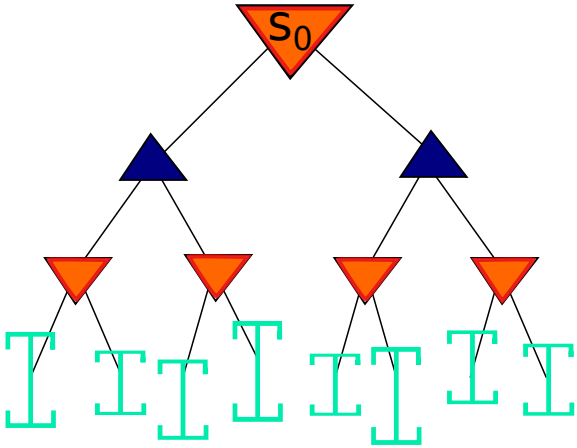
➜ requires confidence intervals on the values $(V_s)_{s \in \mathcal{C}_0}$

➜ 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 $\ell \in \mathcal{L}$,

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

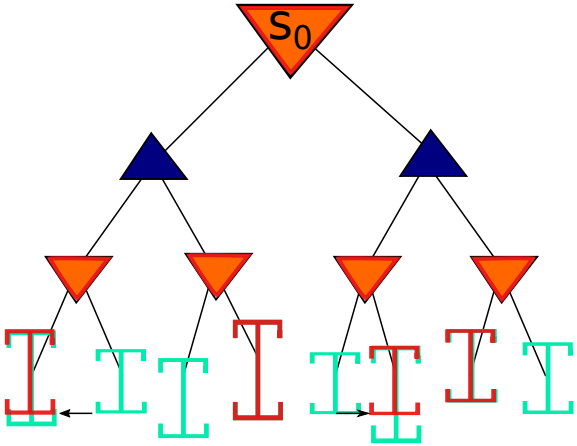


Idea: Propagate these confidence intervals up in the tree

First tool: confidence intervals

MAX node:

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

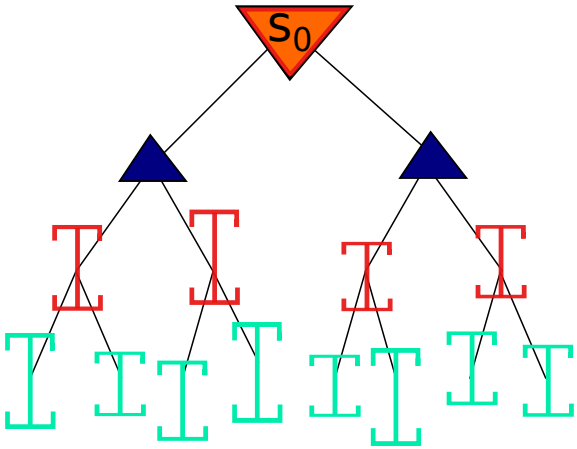


Idea: Propagate these confidence intervals up in the tree

First tool: confidence intervals

MAX node:

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

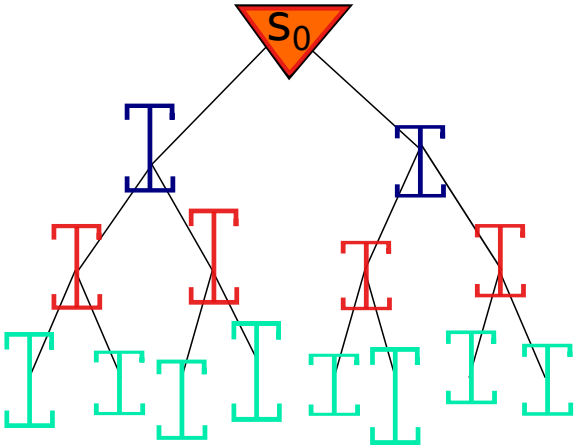


Idea: Propagate these confidence intervals up in the tree

First tool: confidence intervals

MIN node:

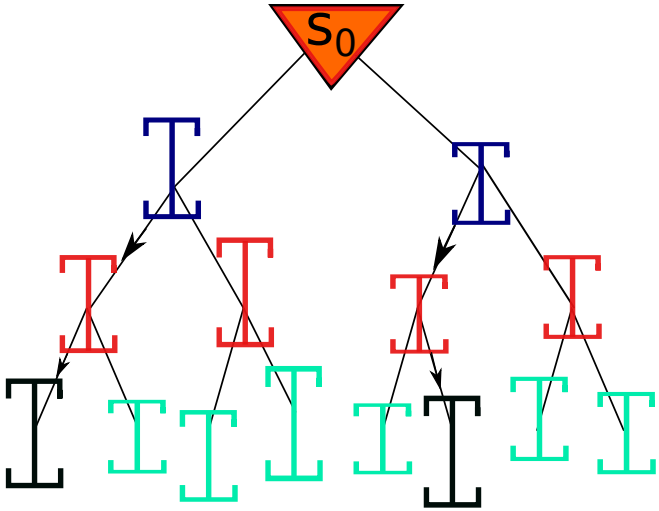
$$\mathrm{UCB}_s(t) = \min_{c \in \mathcal{C}(s)} \mathrm{UCB}_c(t) \qquad \mathrm{LCB}_s(t) = \min_{c \in \mathcal{C}(s)} \mathrm{LCB}_c(t)$$



Idea: Propagate these confidence intervals up in the tree

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

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

- sample the representative leaf associated to that node:

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

($\simeq$ from depth one, run UCT based on *statistically valid* CIs)

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

Theoretical guarantees

For some exploration function β, define

$$\mathrm{LCB}_\ell(t) = \hat{\mu}_\ell(t) - \sqrt{\frac{\beta(N_\ell(t),\delta)}{2N_\ell(t)}},$$

$$\mathrm{UCB}_\ell(t) = \hat{\mu}_\ell(t) + \sqrt{\frac{\beta(N_\ell(t),\delta)}{2N_\ell(t)}}.$$

Theorem [Kaufmann et al., 2018]

Choosing

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

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

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

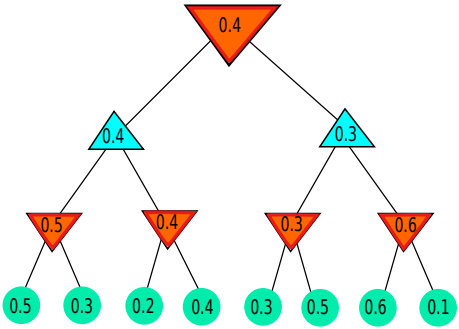
for UGapE-MCTS.

The complexity term

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

where

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



(slightly improved complexity in the work of [Huang et al., 2017])

Future work

- Optimal and efficient algorithms for solving best action identification in a maxmin tree...
- ... and other generic active identification problems
- UCT versus BAI-MCTS on large-scale problem?

Take-home messages

- Best arm identification and regret minimization are two different problems that require different sampling rules
- Upper and Lower Confidence Bounds are useful in both settings
- Optimal algorithms for BAI are inspired by the lower bounds (cf. structured bandits)
- Tools for BAI $\rightarrow$ more general Active Identification problems

- Bandit tools inspire methods for sequential optimization in large spaces (games trees or continuous spaces)

Monte-Carlo Tree Search

Planning strategy: OLOP

Best-arm identification

Monte-Carlo Graph Search

- MCTS is designed for **huge** state space with large branching factor.
- May also apply to smaller MDPs.
- Also, what if several trajectories lead to the **same** state? Can we improve planning ?

Inria

GROUPE RENAULT

Monte-Carlo Graph Search:

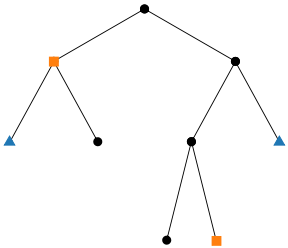
the Value of Merging Similar States

Edouard Leurent[1,2],
Odalric-Ambrym Maillard[1]

[1]Univ. Lille, Inria, CNRS,
Centrale Lille, UMR 9189 – CRIStAL,
[2]Renault Group

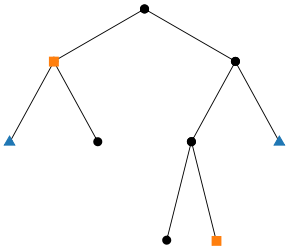
Monte-Carlo Tree Search algorithms

- rely on a tree structure to represent their value estimates.



Monte-Carlo Tree Search algorithms

- rely on a tree structure to represent their value estimates.

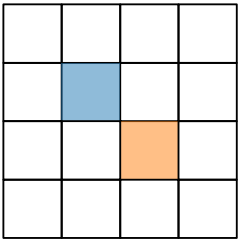


- performance independent of the size S of the state space

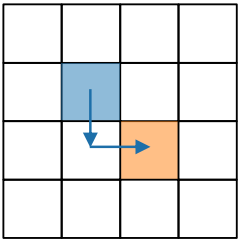
Tabular RL	(UCBVI)	$\sqrt{HSAn}$
MCTS	(OPD)	$n^{-\log \frac{1}{\gamma} / \log A}$

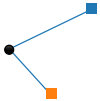
Inria

- There can be several paths to the same state s

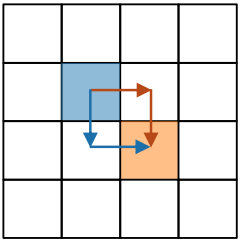


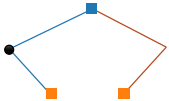
- There can be several paths to the same state s



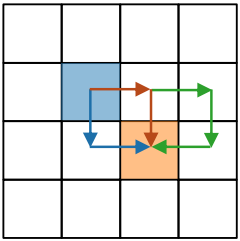


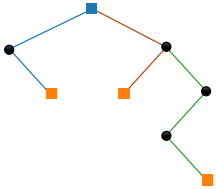
- There can be several paths to the same state s



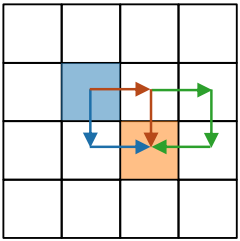


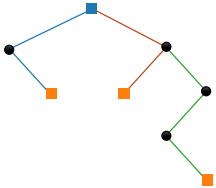
- There can be several paths to the same state s





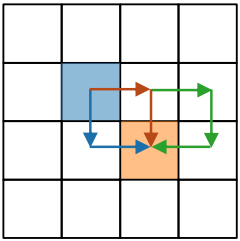
- There can be several paths to the same state s

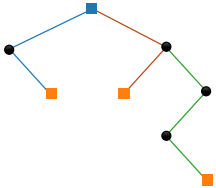




- s is represented several times in the tree

- There can be several paths to the same state s



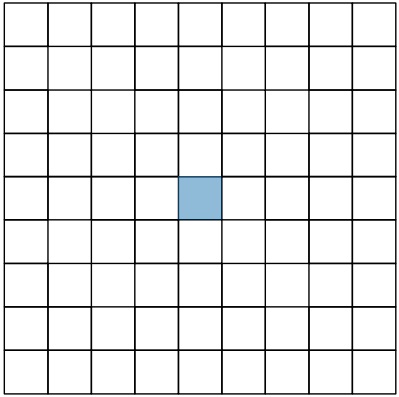


- s is represented several times in the tree
- No information is shared between these paths

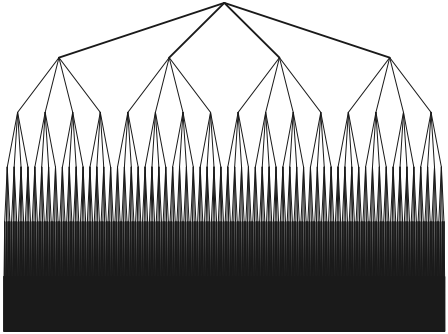
Not accounting for state similarity hinders exploration

Not accounting for state similarity hinders exploration

Sparse gridworld: reward of 0 everywhere



↳ Uniform planning in the space of sequences of actions



OPD, budget of $n = 5460$ moves

Concentration

- Does not lead to uniform exploration of the state space

Concentration

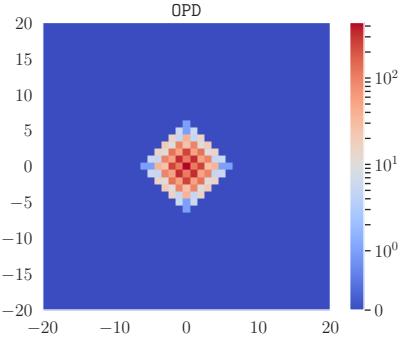
- Does not lead to uniform exploration of the state space
- 2D random walk $\sim$ Rayleigh distribution $P(d) = \frac{2d}{H} e^{-\frac{-d^2}{H}}$

Concentration

- Does not lead to uniform exploration of the state space
- 2D random walk $\sim$ Rayleigh distribution $P(d) = \frac{2d}{H} e^{-\frac{-d^2}{H}}$

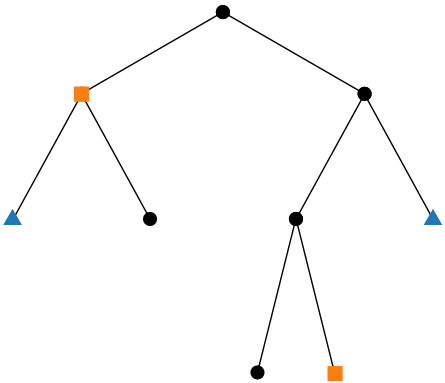
Concentration

- Does not lead to uniform exploration of the state space
- 2D random walk $\sim$ Rayleigh distribution $P(d) = \frac{2d}{H} e^{-\frac{-d^2}{H}}$



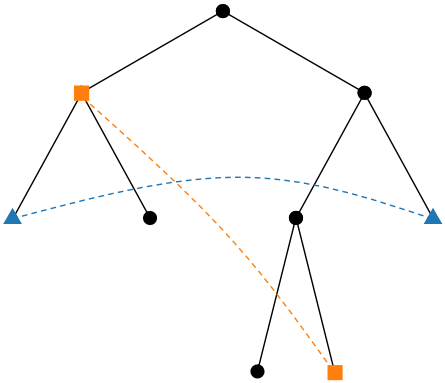
budget of 5460 samples, maximum distance $d = 6$

Better exploit this wasted information



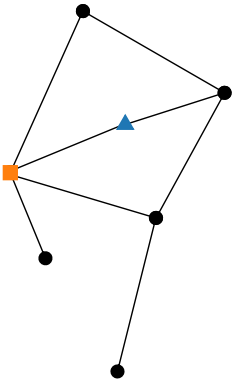
Better exploit this wasted information

- By merging similar states



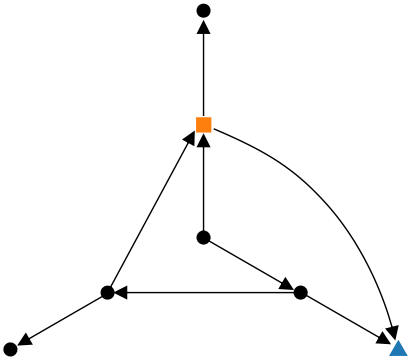
Better exploit this wasted information

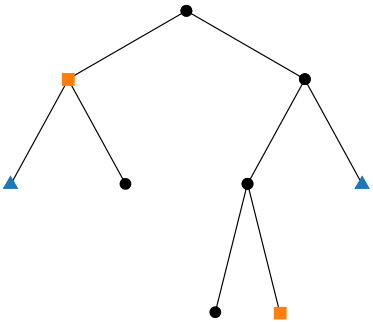
- By merging similar states into a graph

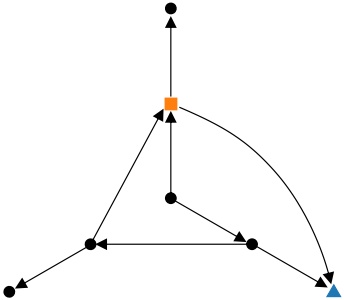


Better exploit this wasted information

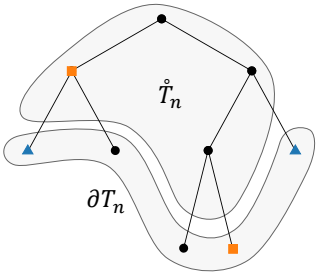
- By merging similar states



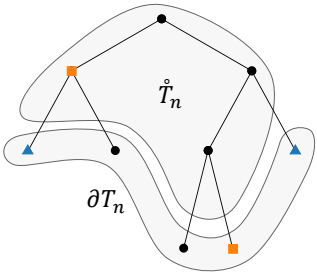




- How to adapt MCTS algorithms to work on graphs?
- Can we quantify the benefit of using graphs over trees?



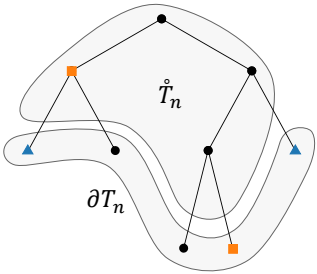
Optimism in the Face of Uncertainty: OPD



Optimism in the Face of Uncertainty: OPD

1. Build confidence bounds $L(a) \leq V(a) \leq U(a)$

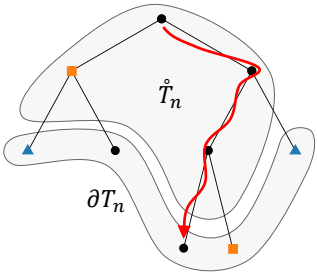
Inria



Optimism in the Face of Uncertainty: OPD

1. Build confidence bounds $L(a) \leq V(a) \leq U(a)$
2. Follow optimistic actions, from the root down to a leaf b

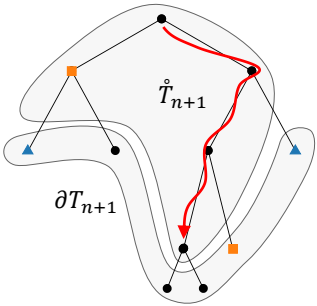
Inria



Optimism in the Face of Uncertainty: OPD

1. Build confidence bounds $L(a) \leq V(a) \leq U(a)$
2. Follow optimistic actions, from the root down to a leaf b
3. Expand the leaf $b \in \partial T_n$

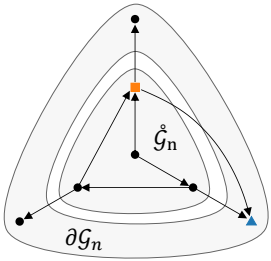
Inria



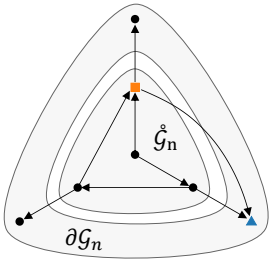
Optimism in the Face of Uncertainty: OPD

1. Build confidence bounds $L(a) \leq V(a) \leq U(a)$
2. Follow optimistic actions, from the root down to a leaf b
3. Expand the leaf $b \in \partial T_n$

Inria

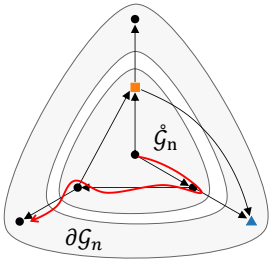


Same principle: GBOP-D



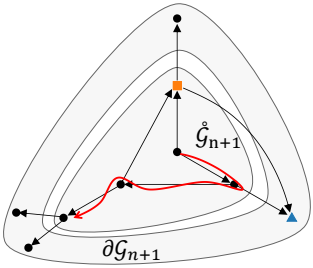
Same principle: GBOP-D

1. Build confidence bounds $L(s) \leq V(s) \leq U(s)$



Same principle: GBOP-D

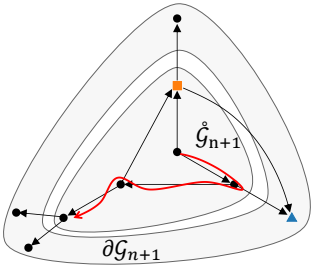
1. Build confidence bounds $L(s) \leq V(s) \leq U(s)$
2. Follow optimistic actions until an external node s is reached



Same principle: GBOP-D

1. Build confidence bounds $L(s) \leq V(s) \leq U(s)$
2. Follow optimistic actions until an external node s is reached
3. Expand the external node $s \in \partial\mathcal{G}_n$

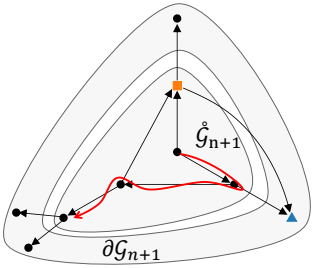
Inria



Same principle: GBOP-D

1. Build confidence bounds $L(s) \leq V(s) \leq U(s)$
2. Follow optimistic actions until an external node s is reached
3. Expand the external node $s \in \partial\mathcal{G}_n$
   > We are guaranteed to expand any state only once.

Inria



Same principle: GBOP-D

1. Build confidence bounds $L(s) \leq V(s) \leq U(s)$
2. Follow optimistic actions until an external node s is reached
3. Expand the external node $s \in \partial\mathcal{G}_n$
   > We are guaranteed to expand any state only once.

Inria

How to bound $V(s) = \sup \sum_{t=0}^{\infty} \gamma^t r_t$?

- Initialize with trivial bounds: $L = 0, \quad U = \frac{1}{1-\gamma}$

How to bound $V(s) = \sup \sum_{t=0}^{\infty} \gamma^t r_t$?

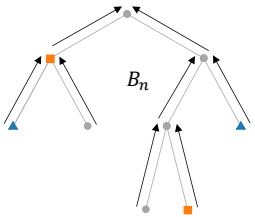
- Initialize with trivial bounds: $L = 0, \quad U = \frac{1}{1-\gamma}$
- Apply the Bellman operator $B : V \rightarrow max_a R(s,a) + \gamma V(s')$

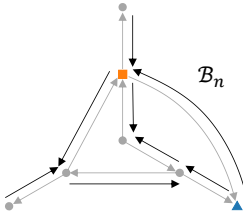
$$L(a) \leq B(L)(a) \leq V(a) \leq B(U)(a) \leq U(a)$$

How to bound $V(s) = \sup \sum_{t=0}^{\infty} \gamma^t r_t$?

- Initialize with trivial bounds: $L = 0, \quad U = \frac{1}{1-\gamma}$
- Apply the Bellman operator $B : V \rightarrow max_a R(s,a) + \gamma V(s')$

$$L(a) \leq B(L)(a) \leq V(a) \leq B(U)(a) \leq U(a)$$

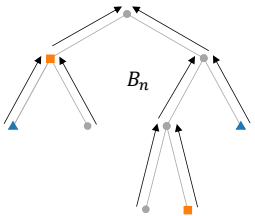


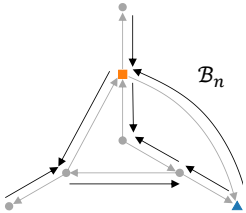


How to bound $V(s) = \sup \sum_{t=0}^{\infty} \gamma^t r_t$?

- Initialize with trivial bounds: $L = 0, \quad U = \frac{1}{1-\gamma}$
- Apply the Bellman operator $B : V \to max_a R(s,a) + \gamma V(s')$

$$L(a) \leq B(L)(a) \leq V(a) \leq B(U)(a) \leq U(a)$$





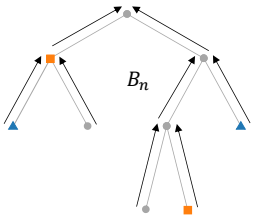
- **Trees**: Converges in d_n steps

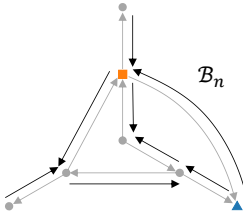
Inria

How to bound $V(s) = \sup \sum_{t=0}^{\infty} \gamma^t r_t$?

- Initialize with trivial bounds: $L = 0, \quad U = \frac{1}{1-\gamma}$
- Apply the Bellman operator $B : V \to max_a R(s,a) + \gamma V(s')$

$$L(a) \leq B(L)(a) \leq V(a) \leq B(U)(a) \leq U(a)$$





- **Trees**: Converges in d_n steps
- **Graphs**: May converge in ∞ steps when there is a loop

Inria

Is GBOP-D more efficient than OPD?

Is GBOP-D more efficient than OPD?
Performance: $r_n = V^\star - V(a_n)$

Is GBOP-D more efficient than OPD?
Performance: $r_n = V^\star - V(a_n)$

Theorem (Sample complexity of OPD, Hren and Munos, 2008)

$$r_n = \widetilde{\mathcal{O}}\left(n^{-\log \frac{1}{\gamma}/\log \kappa}\right),$$

where κ is a problem-dependent difficulty measure.

Is GBOP-D more efficient than OPD?
Performance: $r_n = V^\star - V(a_n)$

Theorem (Sample complexity of OPD, Hren and Munos, 2008)

$$r_n = \widetilde{\mathcal{O}}\left(n^{-\log \frac{1}{\gamma}/\log \kappa}\right),$$

where κ is a problem-dependent difficulty measure.

Theorem (Sample complexity of GBOP-D)

$$r_n = \widetilde{\mathcal{O}}\left(n^{-\log \frac{1}{\gamma}/\log \kappa_\infty}\right),$$

where κ_∞ is a tighter problem-dependent difficulty measure:

$$\kappa_\infty \leq \kappa$$

Inria

- $\kappa_\infty = \kappa$ if the MDP has a tree structure

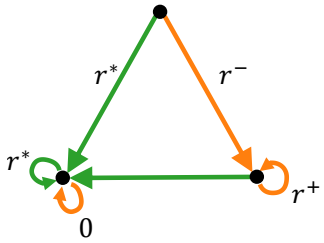
- $\kappa_\infty = \kappa$ if the MDP has a tree structure
- $\kappa_\infty < \kappa$ when trajectories intersect a lot

- $\kappa_\infty = \kappa$ if the MDP has a tree structure
- $\kappa_\infty < \kappa$ when trajectories intersect a lot
  - actions cancel each-other out (e.g. moving left or right)

- $\kappa_\infty = \kappa$ if the MDP has a tree structure
- $\kappa_\infty < \kappa$ when trajectories intersect a lot
  - actions cancel each-other out (e.g. moving left or right)
  - actions are commutative (e.g. placing pawns on a board)

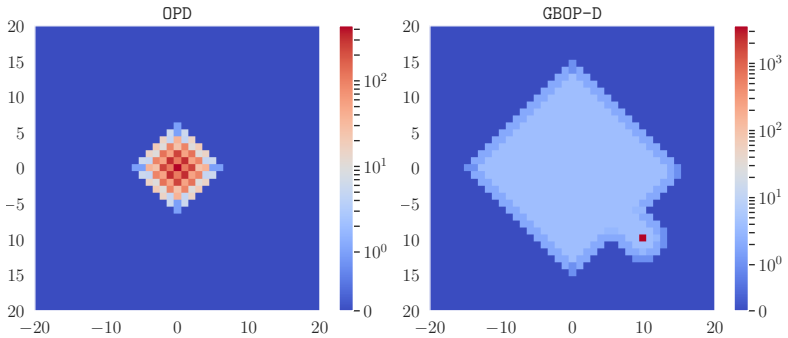
- $\kappa_\infty = \kappa$ if the MDP has a tree structure
- $\kappa_\infty < \kappa$ when trajectories intersect a lot
  - > actions cancel each-other out (e.g. moving left or right)
  - > actions are commutative (e.g. placing pawns on a board)

Illustrative example: 3 states, $K > 2$ actions



$$\kappa_\infty = 1 < \kappa = K - 1$$

Rewards in a ball around $(10, 10)$ of radius 5, with quadratic decay

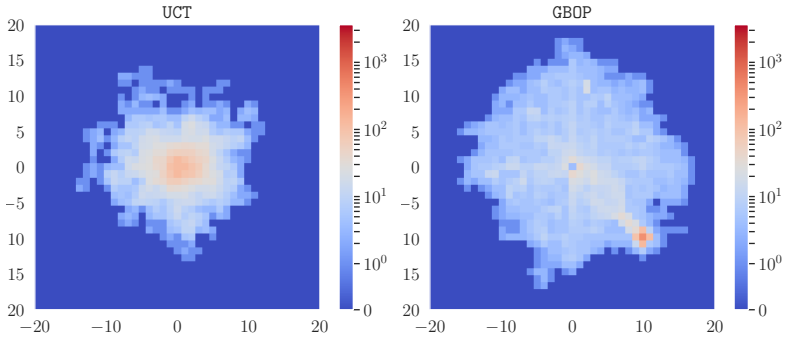


$n = 5640$ samples

Extension to stochastic MDPs

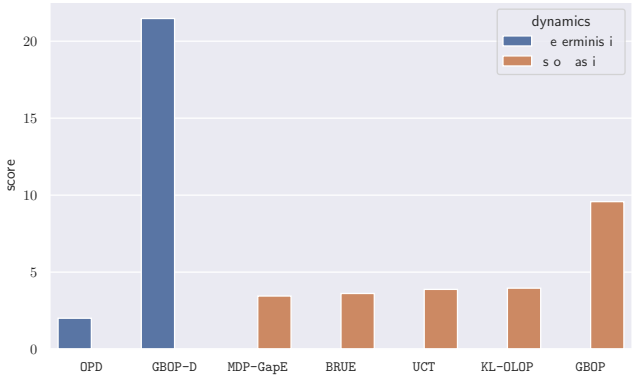
- Use state similarity to tighten the bounds $L \leq V \leq U$.
- We adapt `MDP-GapE` (Jonsson et al., 2020) to obtain `GBOP`

Noisy transitions with probability $p = 10\%$



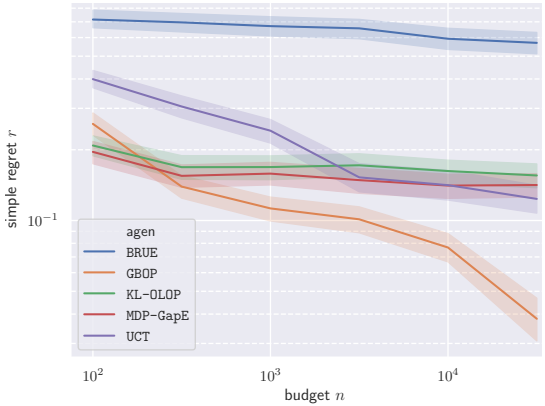
$n = 5640$ samples

$$S = \sum_{t=1}^{n} \underbrace{d(s_t, s_0)}_{\text{Exploration}} - \underbrace{d(s_t, s_g)}_{\text{Exploitation}}$$

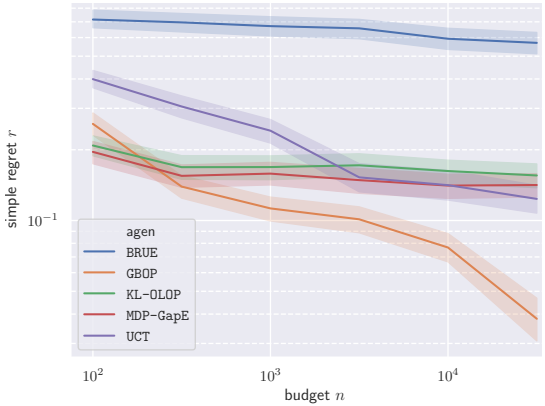


$n = 5640$ samples

Sailing Domain (Vanderbei, 1996)



Sailing Domain (Vanderbei, 1996)



Effective branching factor κ_e:

- $\kappa_e \approx 3.6$, for `BRUE`, `KL-OLOP`, `MDP-GapE`, `UCT`
- $\kappa_e \approx 1.2$ for `GBOP`, which suggests our results may still hold

"Homework"

- Consider AlphaZero (MCTS+Deep-Learning)
- Replace MCTS with MCGS
- Combine with BAI
- Replace UCT mechanism with KL-OLOP mechanism.

Self-check on planning

- Monte Carlo Tree Search
- What are the 4 main steps of of MCTS strategy?
- UCT rule for the value of each node.
- What is a Generative model?
- KL-OLOP combines two main algorithms: which ones?
- What is Best-armed identification (BAI) objective?
- What is Simple regret?
- Fixed-budget objective vs Fixed-confidence objective
- Reduction from cumulative to simple regret
- Sequential Halving
- What do we track in Track-and-stop?
- What is forced exploration?
- UCT rule in max node, versus UCT rule in min node.
- Monte-Carlo Graph Search idea
- When to rather use MGTS? When to rather use MCTS?

"*The more applied you go, the stronger theory you need*"

Merci

odalricambrym.maillard@inria.fr
odalricambrymmaillard.wordpress.com