Reinforcement learning and bandits

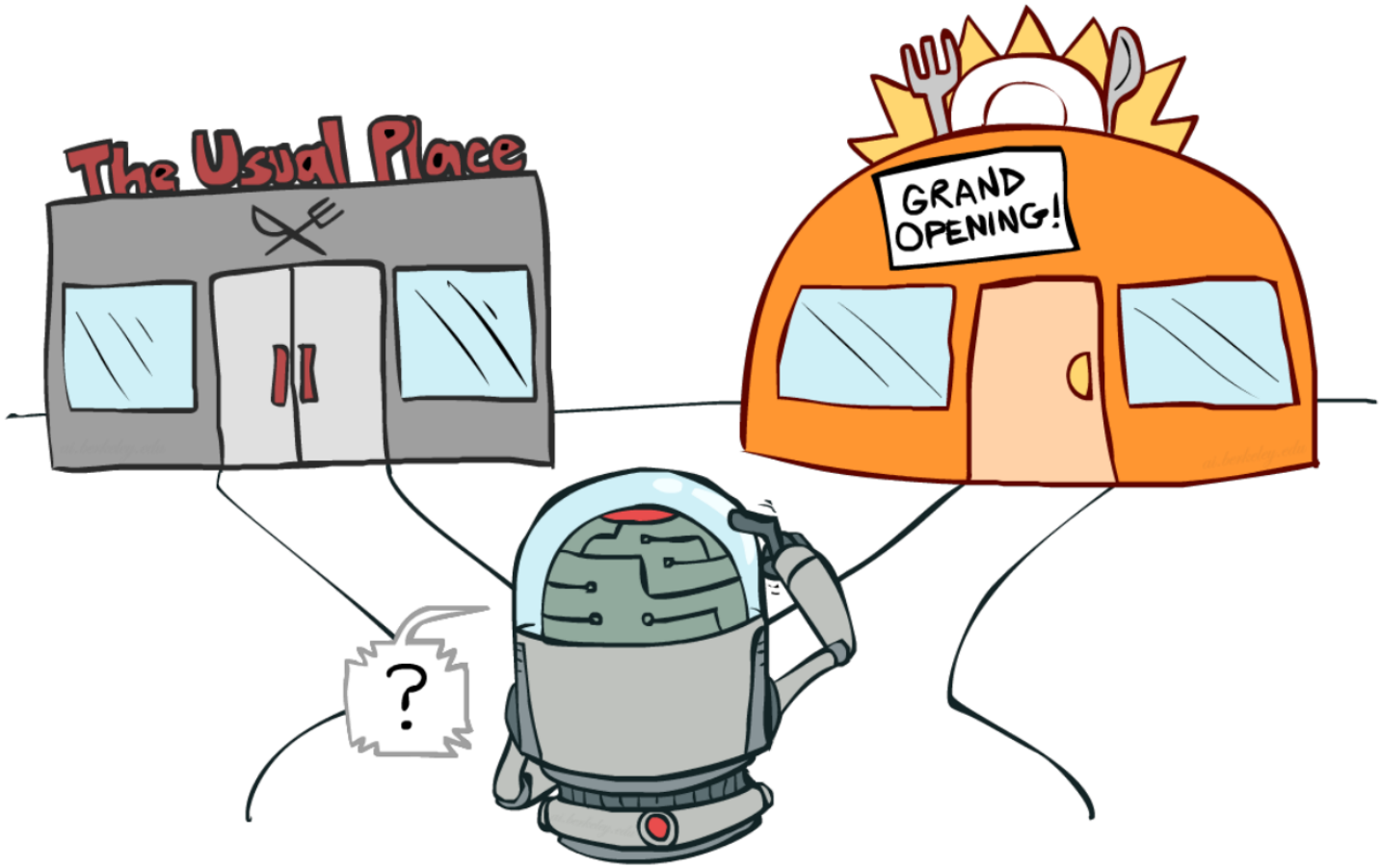
Exploration-Exploitation Tradeoffs

Nicolas Gast

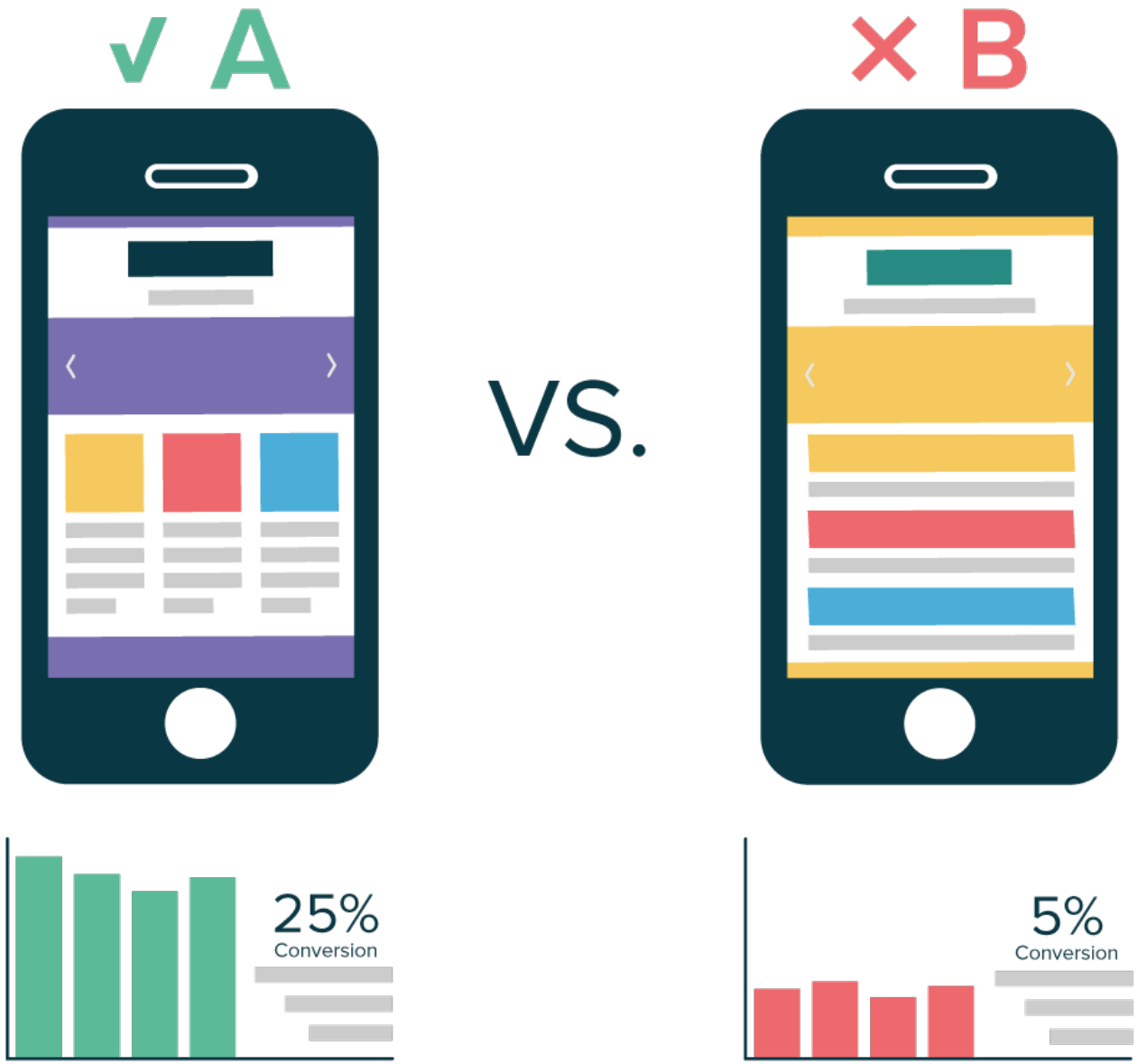
Inria

Séminaire LIG – Ens de Rennes, janvier 2022

The exploration-exploitation dilemma



Example: AB-testing



Example: move exploration in games

How to decide when and what to explore?

We need models

- To think of the tradeoff.
- To design new algorithms.

Outline

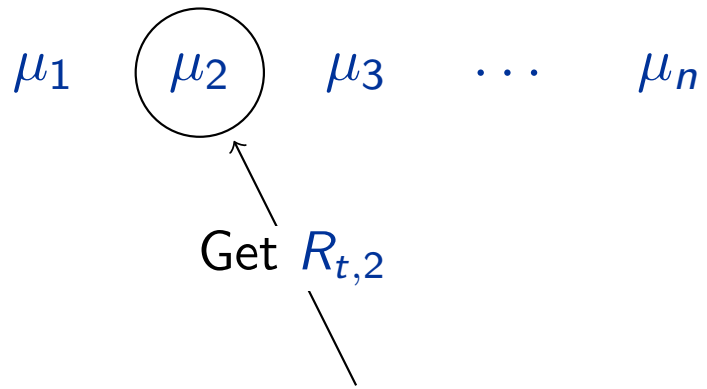
1. Stochastic bandits and regret
   - Definition of regret
   - The UCB algorithm
2. Monte-Carlo Tree Search
   - Min-max and alpha-beta pruning
   - MCTS and exploration
3. Conclusion

Outline

1. Stochastic bandits and regret
   - Definition of regret
   - The UCB algorithm
2. Monte-Carlo Tree Search
   - Min-max and alpha-beta pruning
   - MCTS and exploration
3. Conclusion

The Bernoulli multi-armed bandit

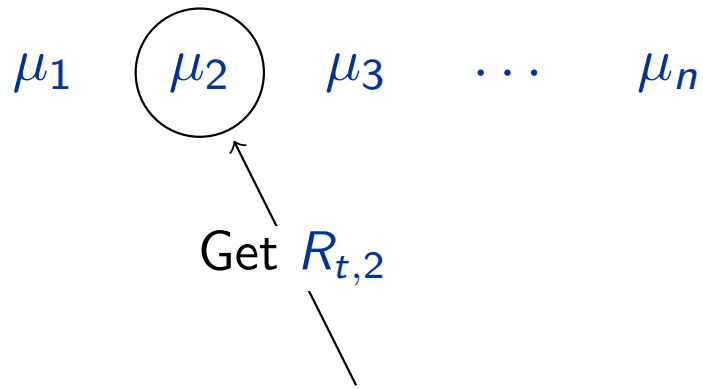
At each time step, you make a choice $A_t \in \{1 \dots n\}$.



The average reward of a is $\mathbb{E}[R_{t,a}] = \mu_a$ but you do not know the μ_as.

The Bernoulli multi-armed bandit

At each time step, you make a choice $A_t \in \{1 \dots n\}$.



The average reward of a is $\mathbb{E}[R_{t,a}] = \mu_a$ but you do not know the μ_as.

Assumption: The rewards are independent and Bernoulli.

This is called stochastic Bernoulli bandit.

Motivation

- **Maximize clicks**, e.g., the choice of a title of a news article :

Title	Click proba.
"Murder victim found in adult entertainment venue"	μ_1
"Headless Body found in Topless Bar"	μ_2

  - Choose which title to display. Observe (click or no click).

This is close to **A-B testing**.

Motivation

- **Maximize clicks**, e.g., the choice of a title of a news article :

Title	Click proba.
"Murder victim found in adult entertainment venue"	μ_1
"Headless Body found in Topless Bar"	μ_2

  - Choose which title to display. Observe (click or no click).

This is close to **A-B testing**.

- **Clinical trial**

μ_1 μ_2 μ_3 μ_4




  - Choose treatment A_t for patient t. Observe healed / not healed.

Our metric is the regret

If you know the values of μ_as, you should pick $\arg\max_a \mu_a$.

Our metric is the regret

If you know the values of μ_as, you should pick $\arg\max_a \mu_a$.

We define the regret of a sequence of action $\mathcal{A} = (A_1, A_2 \dots)$ as

$$\text{Regret}(\mathcal{A}, T) = \underbrace{(T \max_a \mu_a)}_{\text{reward of an oracle}} - \underbrace{\mathbb{E}\left[\sum_{t=1}^{T} R_{t,A_t}\right]}_{\text{your reward}}.$$

- Goal : design strategies that have a small regret (regardless of μ).

Some ideas of policies

- **Random** – Draw each arm with probability $1/n$.
  - Exploration

Some ideas of policies

- **Random** – Draw each arm with probability $1/n$.
  - Exploration

- **Greedy:** Always choose the empirical best arm:

$$A_{t+1} = \operatorname*{arg\,max}_{a \in \{1 \dots n\}} \hat{\mu}_a(t)$$

  - Exploitation

Some ideas of policies

- **Random** – Draw each arm with probability $1/n$.
  - Exploration
- **Greedy:** Always choose the empirical best arm:

$$A_{t+1} = \arg\max_{a \in \{1 \dots n\}} \hat{\mu}_a(t)$$

  - Exploitation
- ε-**greedy** : apply "greedy" with probability $1 - \varepsilon$ and "random" otherwise (each with probability ε/n)
  - Exploration and exploitation.

Some ideas of policies

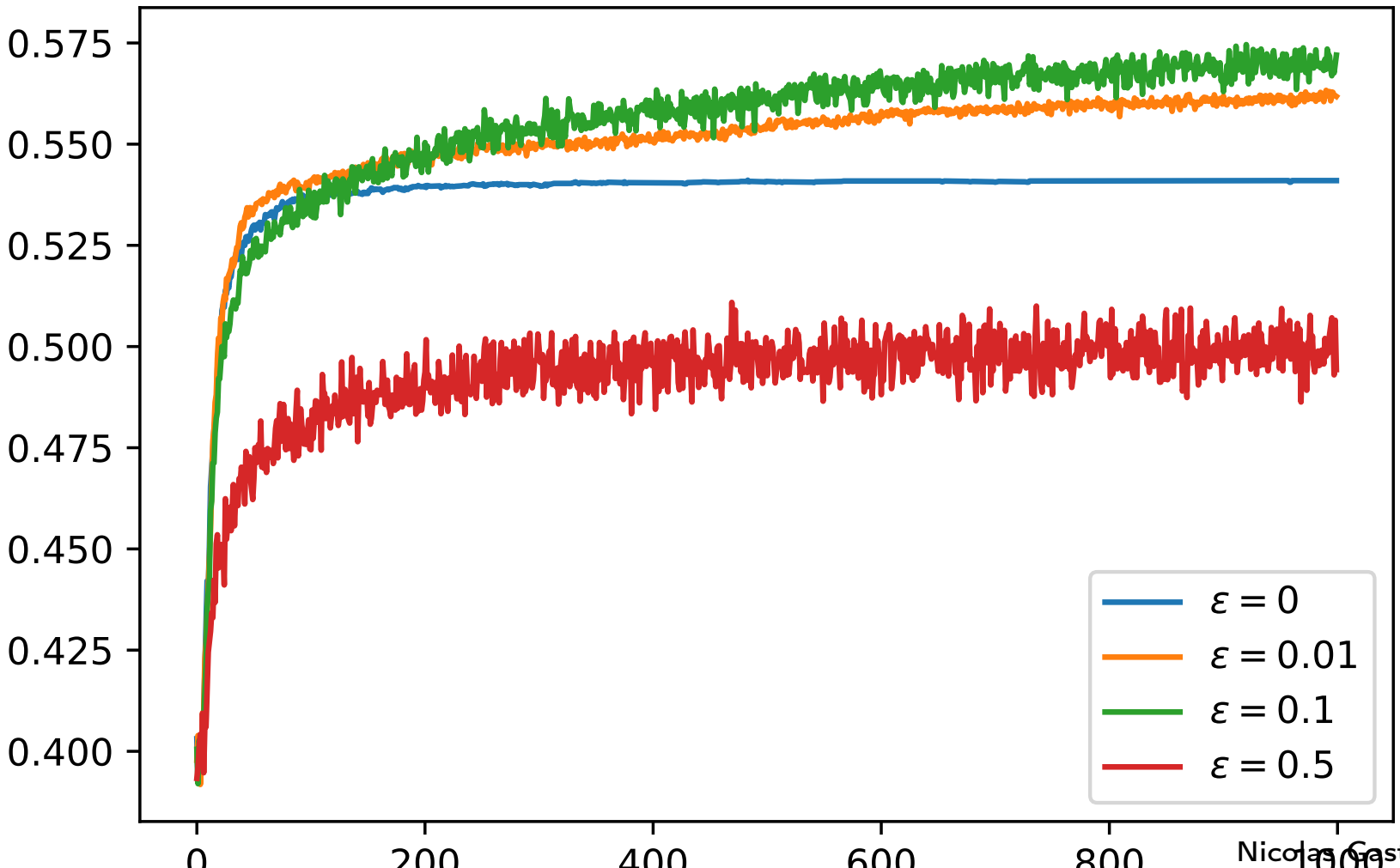
- **Random** – Draw each arm with probability $1/n$.
  - Exploration
- **Greedy:** Always choose the empirical best arm:

$$A_{t+1} = \arg\max_{a \in \{1 \dots n\}} \hat{\mu}_a(t)$$

  - Exploitation
- ε-**greedy** : apply "greedy" with probability $1 - \varepsilon$ and "random" otherwise (each with probability ε/n)
  - Exploration and exploitation.

ε-greedy : Smaller or larger ϵ are not necessarily better

$$\text{Regret}(\varepsilon\text{-greedy}, T) = T(\sum_{a=1}^{n}(\mu_* - \mu_a))\frac{\varepsilon}{n} + o(T) \text{ if } \varepsilon > 0.$$



Asymptotically optimal regret

ε-greedy policies have $O(T)$ **regret** (this is called linear regret).

Can we do better?

[1] Meaning Regret$(\mathcal{I}, T) = o(T^{\alpha})$ for all μ and α.

Asymptotically optimal regret

ε-greedy policies have $O(T)$ **regret** (this is called linear regret).

Can we do better?

Theorem (Lai and Robbins, 1985. (Asymptotically Efficient Adaptive Allocation Rules))
There exists a constant c (that depends on μ) such that any uniformly efficient[1] *strategy satisfies :*

$$Regret(\mathcal{A}, T) \geq c \log T$$

[1] Meaning $\text{Regret}(\mathcal{I}, T) = o(T^{\alpha})$ for all μ and α.

UCB builds on Confidence Intervals

Consider a coin that gives "Head" with probability μ. Suppose that you draw a coin N times and observe K times "head". The natural estimator of μ is:

$$\hat{\mu} = \frac{K}{N}$$

UCB builds on Confidence Intervals

Consider a coin that gives "Head" with probability μ. Suppose that you draw a coin N times and observe K times "head". The natural estimator of μ is:

$$\hat{\mu} = \frac{K}{N}$$

Hoeffding inequality gives us

$$\mathbb{P}\left(\hat{\mu} - \sqrt{\frac{\alpha}{2N}} \leq \underbrace{\mu}_{\text{real } \mu} \leq \underbrace{\hat{\mu} + \sqrt{\frac{\alpha}{2N}})}_{\text{upper confidence bound}} \right) \geq 1 - 2e^{-\alpha}.$$

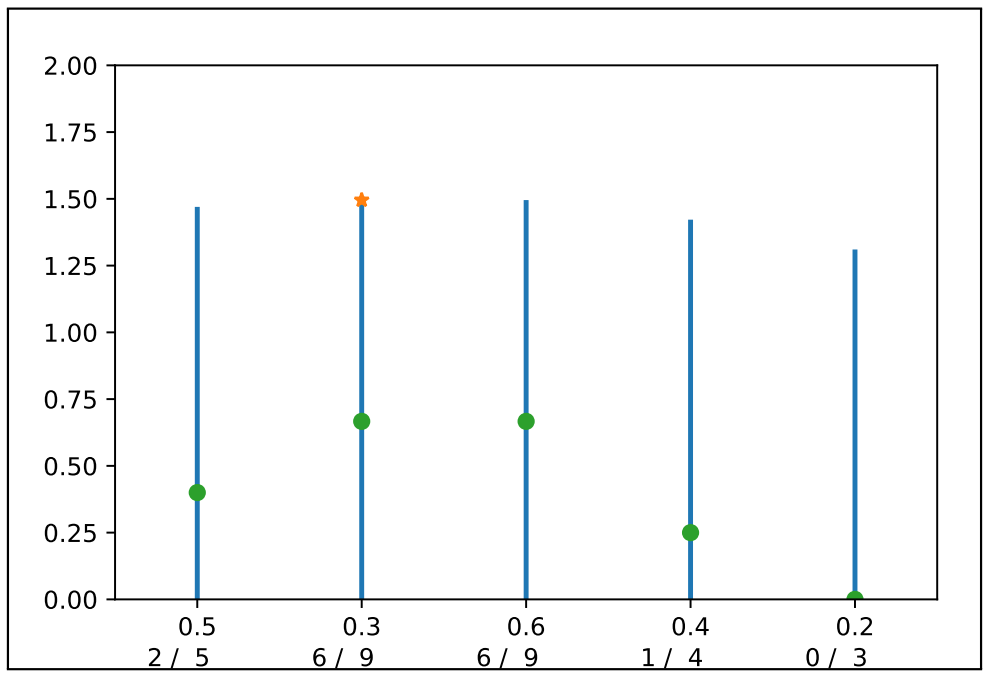
The idea of UCB is to use the above bound with a growing α.

The UCB algorithm

UCB computes a confidence bound $UCB_a(t)$ such that $\mu_a(t) \leq UCB_a(t)$ with high probability. Example : $UCB1$ [Auer et al. 02] uses

$$UCB_a(t) = \hat{\mu}_a(t) + \sqrt{\frac{\alpha \log t}{2N_a(t)}}.$$

- Choose $A_{t+1} \in \arg\max_{a \in \{1 \dots n\}} UCB_a(t)$ (optimism principle).



UCB has logarithmic regret

Theorem. $Regret(UCB) \leq c \log(T)$.

UCB is an OFU algorithm: Optimism in the Face of Uncertainty.

UCB has logarithmic regret

Theorem. $Regret(UCB) \leq c \log(T)$.

UCB is an OFU algorithm: Optimism in the Face of Uncertainty.

Idea of optimism. Let $\tilde{\mu}_a = \hat{\mu}_{t,a} + bonus_{t,a}$. Note that $\tilde{\mu}_{A_t} = \max_a \tilde{\mu}_a$.

UCB has logarithmic regret

Theorem. $Regret(UCB) \leq c \log(T)$.

UCB is an OFU algorithm: Optimism in the Face of Uncertainty.

Idea of optimism. Let $\tilde{\mu}_a = \hat{\mu}_{t,a} + bonus_{t,a}$. Note that $\tilde{\mu}_{A_t} = \max_a \tilde{\mu}_a$.

$$
\begin{aligned}
Regret &= \sum_t \max_a \mu_a - \mu_{A_t} \\
&= \sum_t \underbrace{\max_a \mu_a - \max_a \tilde{\mu}_a}_{\text{Optimism}} + \underbrace{\tilde{\mu}_{A_t} - \mu_{A_t}}_{\text{Concentration.}}
\end{aligned}
$$

≤ 0 if bonus large.

small if bonus small.

More on bandits

- Bayesian approach (Thompson sampling 1933, analyzed in Kauffman et al 2012)
- What about MDPs?
  - UCRL2, UCBVI,...
- Adversarial aspects, games

Outline

1. Stochastic bandits and regret
   - Definition of regret
   - The UCB algorithm
2. Monte-Carlo Tree Search
   - Min-max and alpha-beta pruning
   - MCTS and exploration
3. Conclusion

Tree search

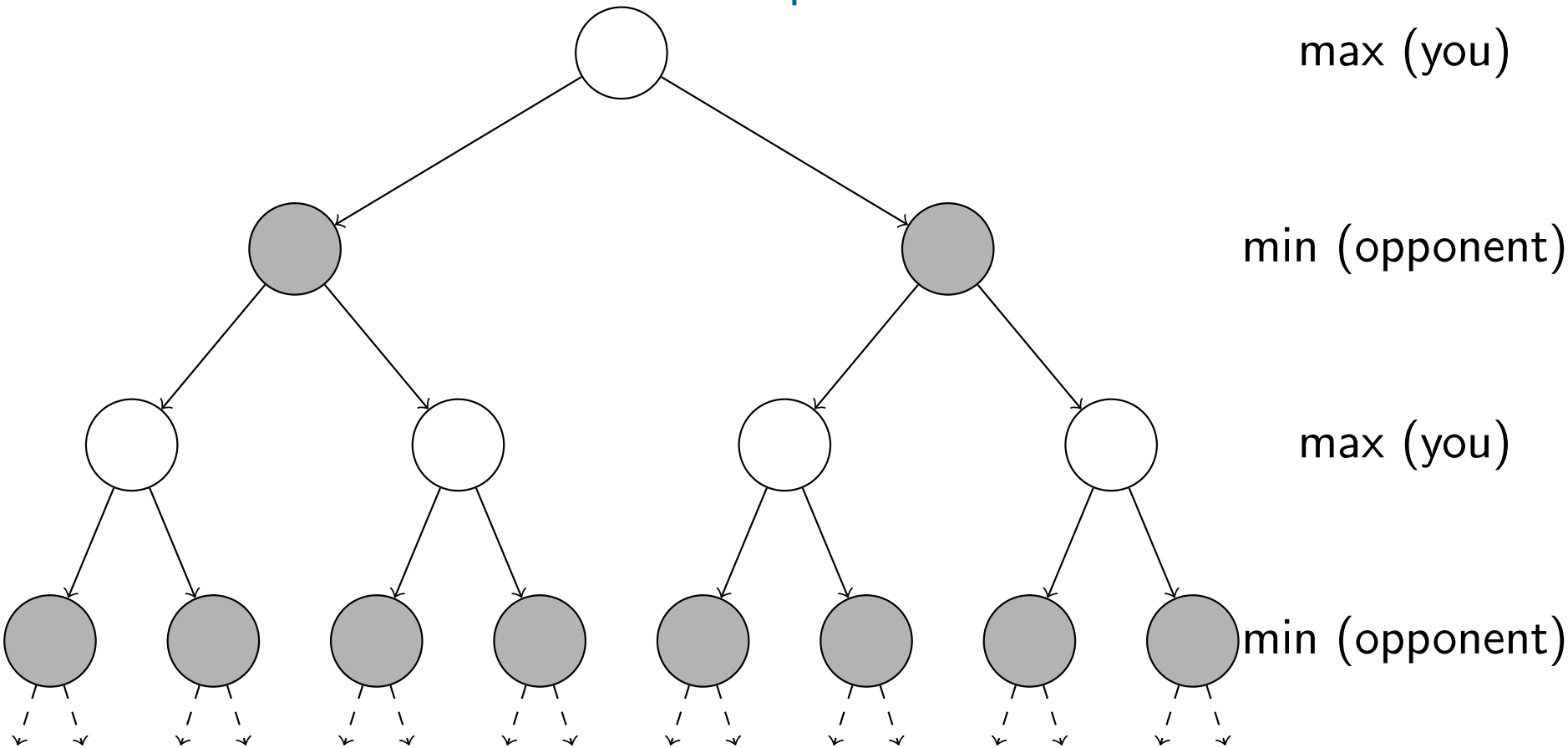
For turn-based two players zero sum games

From a given position, takes the best decision.

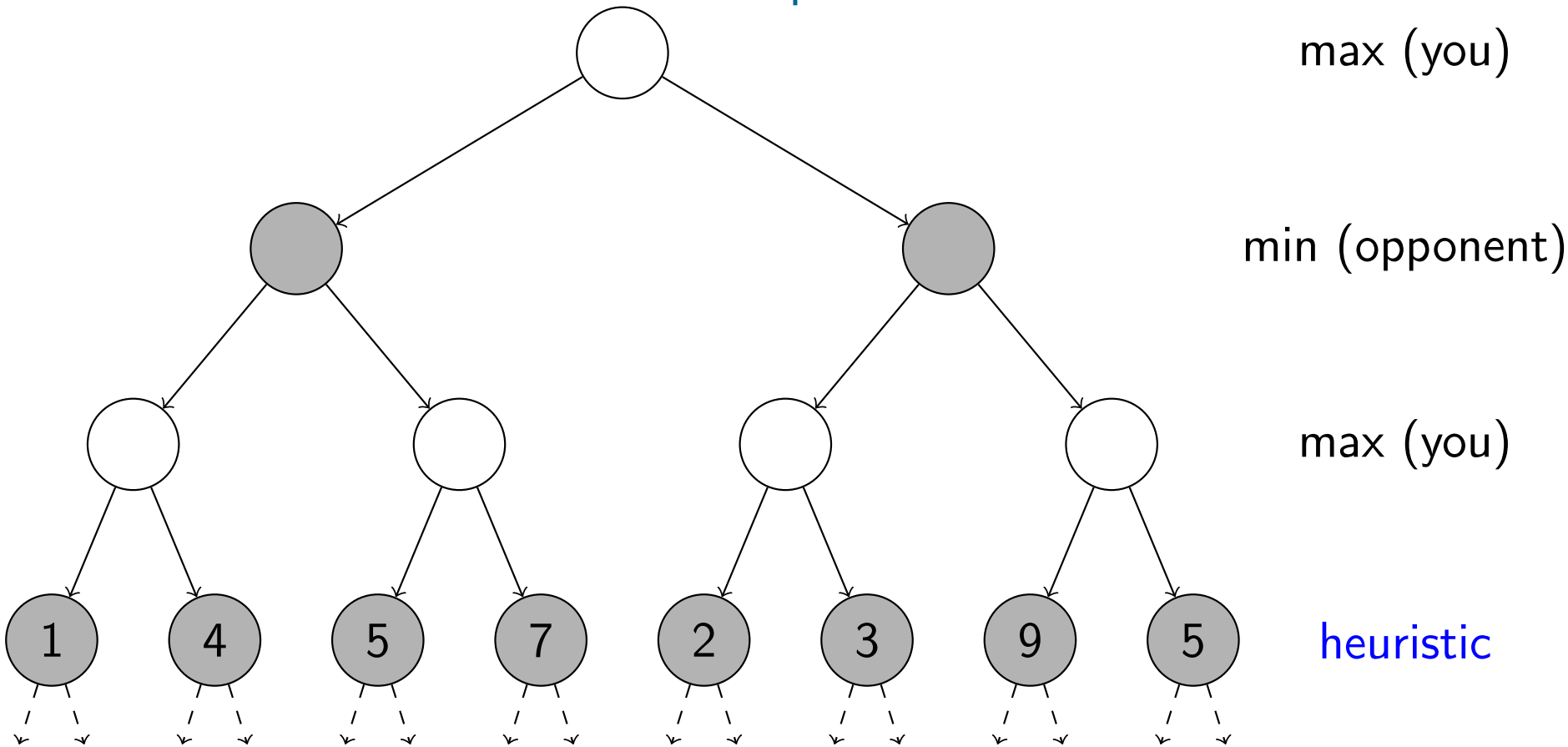
- Generate a tree of possibilities.
- Explore this tree.

What if the tree is too big?

You can construct the tree of possibilities



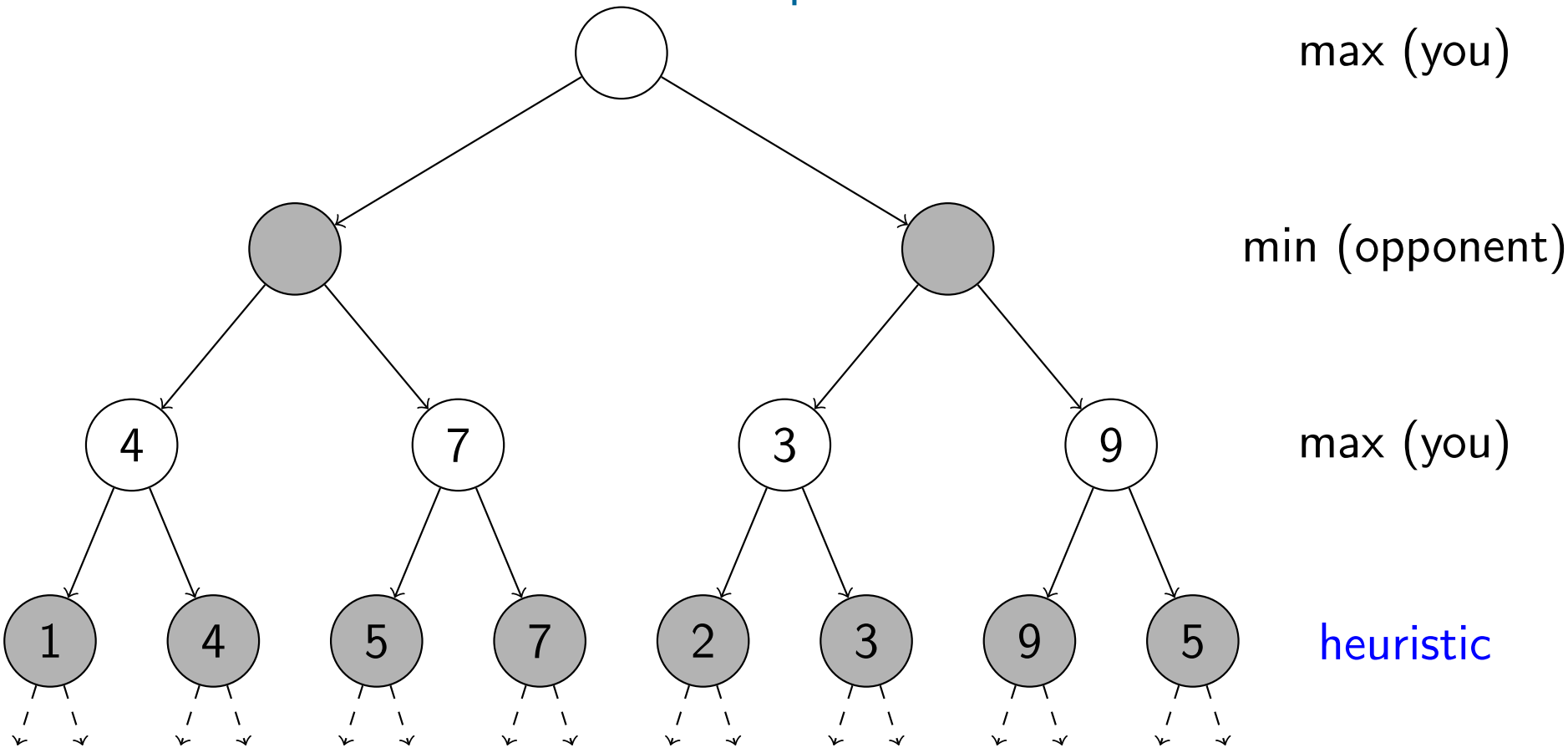
You can construct the tree of possibilities



If the tree is too big, you stop at depth D and use a heuristic.

- You can backtrack with the min-max algorithm.

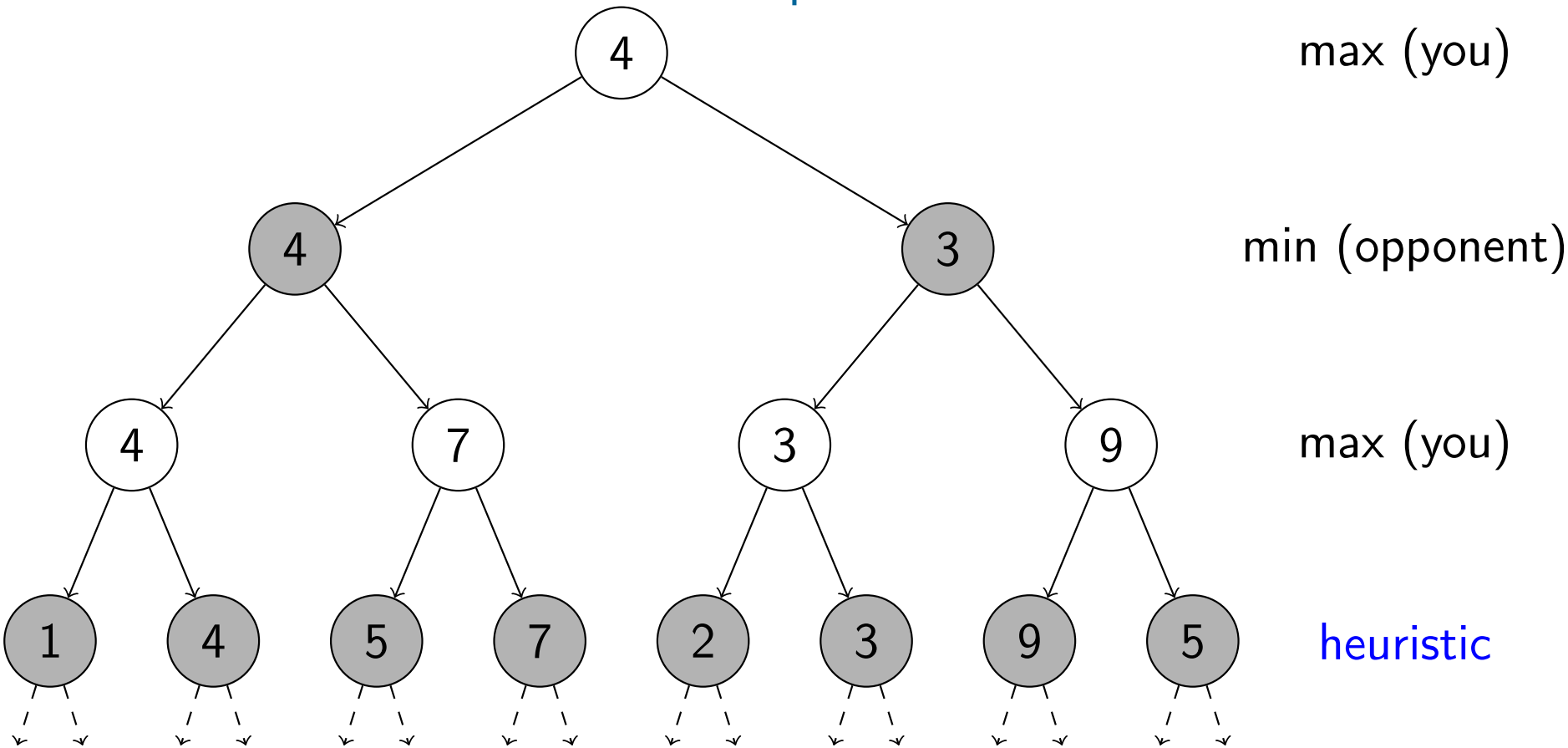
You can construct the tree of possibilities



If the tree is too big, you stop at depth D and use a heuristic.

- You can backtrack with the min-max algorithm.

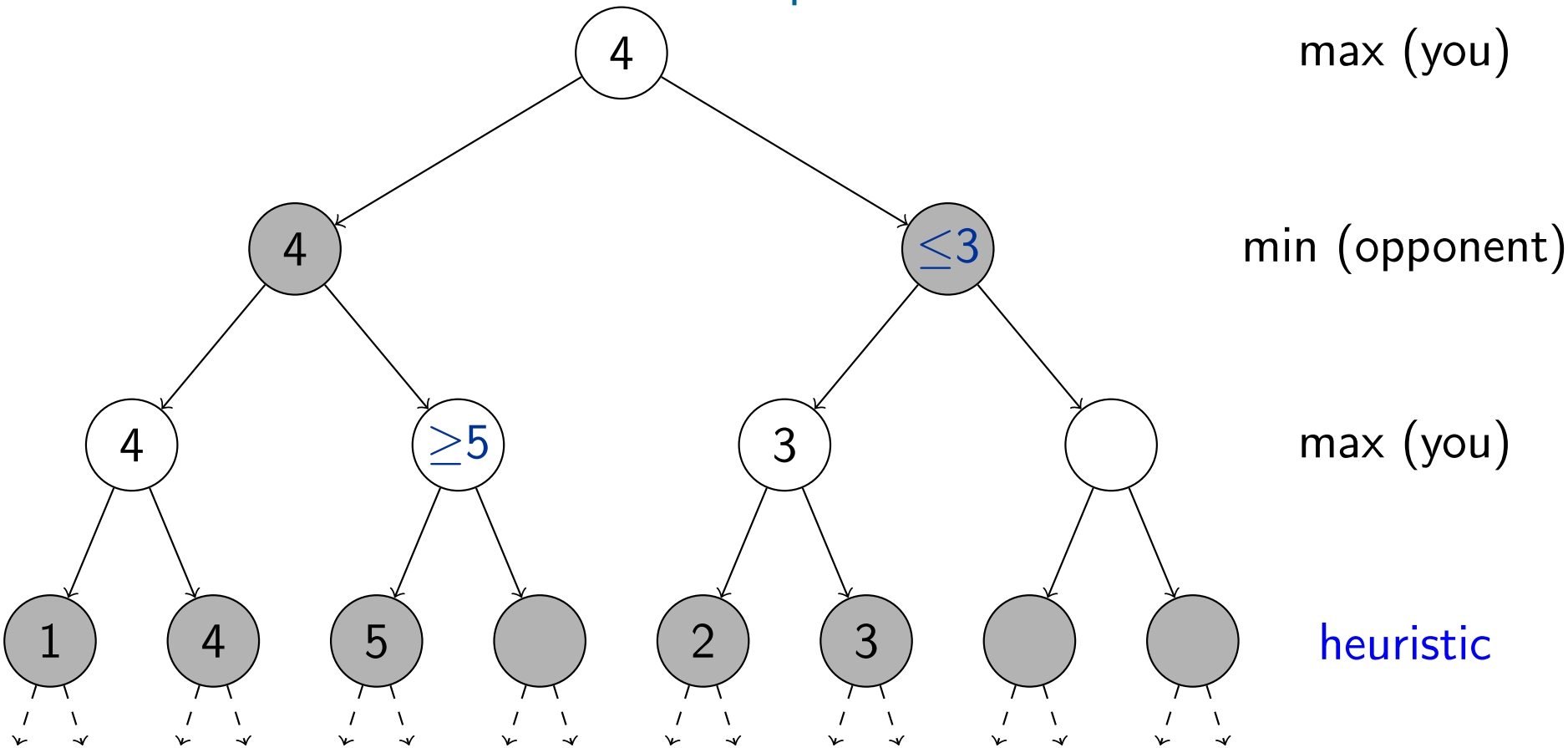
You can construct the tree of possibilities



If the tree is too big, you stop at depth D and use a heuristic.

- You can backtrack with the min-max algorithm.

You can construct the tree of possibilities



If the tree is too big, you stop at depth D and use a heuristic.

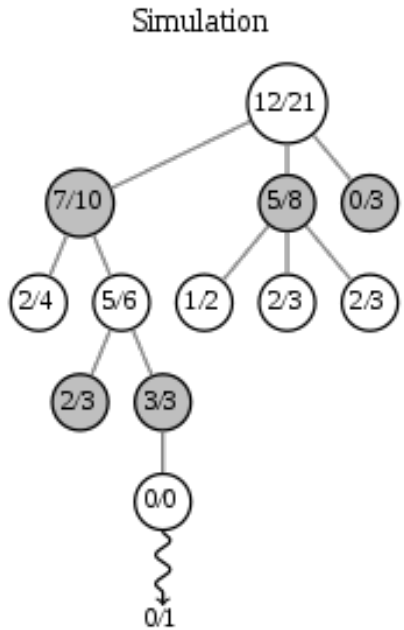
- You can backtrack with the min-max algorithm.
- For optimization, you can use alpha-beta pruning.

Min-max and alpha-beta perform well (ex: Chess)...

...but can be limited (ex: go)

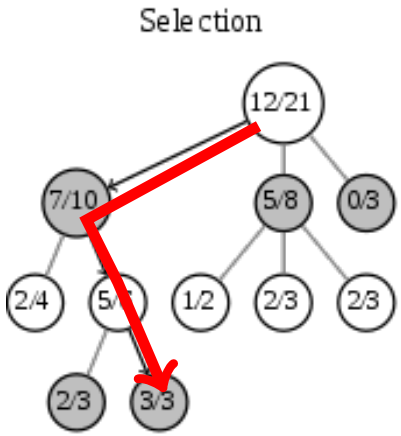
- Tree can still be very big (A^D)
- You need a good heuristic.
  - Result is only available at the end
- You might want to avoid the exploration of not promising parts.
  - For that you need a good heuristic.

MCTS (Monte Carlo Tree Search) uses simulation to conduct the tree search



- Simulate many games and compute how many were won.
- Explore carefully which actions were best.

MCTS (Monte Carlo Tree Search) uses simulation to conduct the tree search

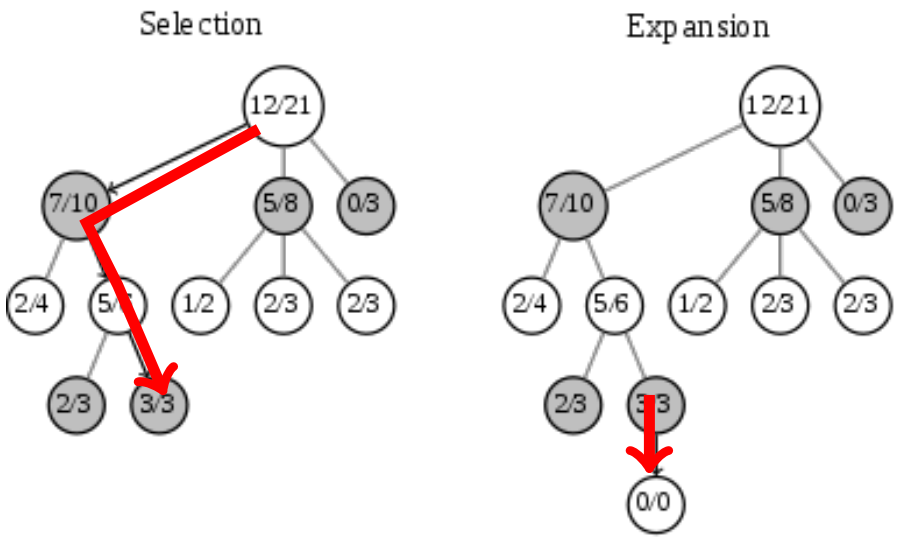


For each child, let $S(c)$ be the number of success and $N(c)$ be the number of time you played c, and $t = \sum_{c'} N(c')$.

- Explore $\arg\max_c \frac{S(c)}{N(c)} + 2\sqrt{\frac{\log t}{N(c)}}$.

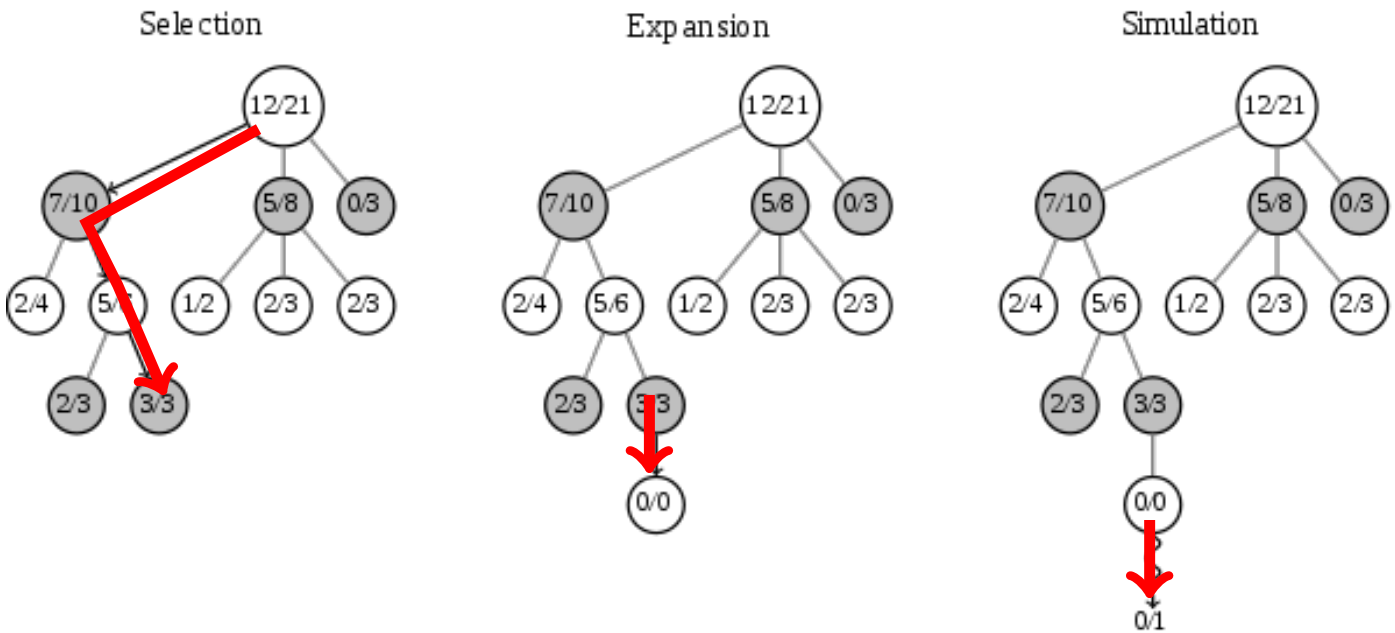
Open question: no guarantee with $\sqrt{\log t / N(c)}$. Is $\sqrt{t}/N(c)$ better?

MCTS (Monte Carlo Tree Search) uses simulation to conduct the tree search



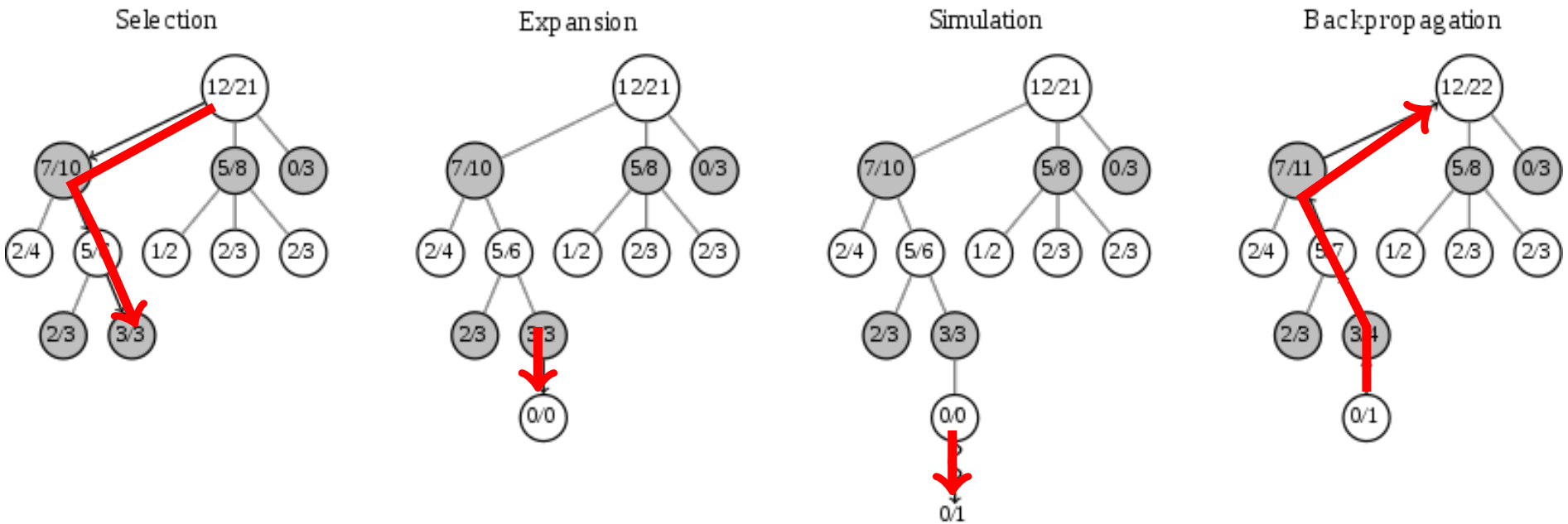
- Create one or multiple children of the leaf.

MCTS (Monte Carlo Tree Search) uses simulation to conduct the tree search



- Obtain a value of the node (e.g. rollout)

MCTS (Monte Carlo Tree Search) uses simulation to conduct the tree search



Rollout policy
(ex: random)

- Backpropagate to the root

MCTS algorithm

MCTS

1: **while** Some time is left **do**
2: Select a leaf node *#UCB-like*
3: Expand a leaf
4: Use rollout (or equivalent) to estimate the leaf *#random sampling*
5: Backpropagate to the root
6: **end while**
7: Return $\arg\max_{c \in \text{children(root)}} N(c)$ *#or* $S(c)/N(c)$.

Outline

1. Stochastic bandits and regret
   - Definition of regret
   - The UCB algorithm
2. Monte-Carlo Tree Search
   - Min-max and alpha-beta pruning
   - MCTS and exploration
3. Conclusion

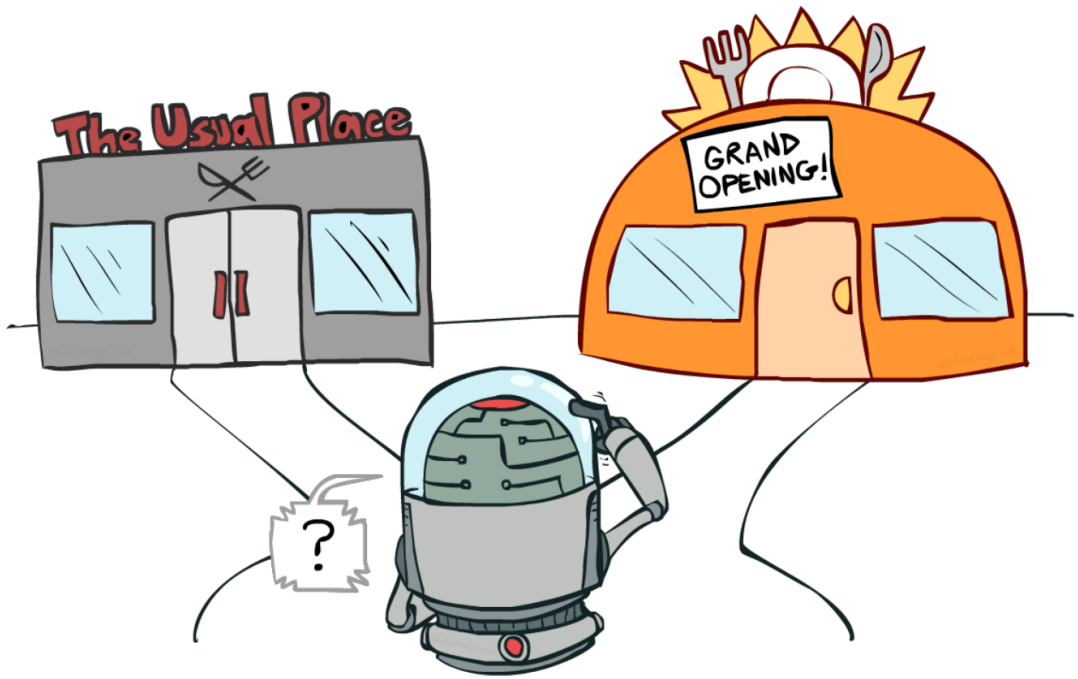
Exploration v.s. exploitation is central in RL

- Bandits and regret help formalizing this idea.
- One important notion is the use of optimism to force exploration.
  - Bayesian sampling can also be used
- Theoretical tools guide practical implementations.

In our team (POLARIS) Design of experiments (choice of hyper-parameters); Markovian bandits: learning and optimization; Stochastic games; Routing algorithms.

To avoid missing good opportunities, retry bad experiences

(but not too much)



http://polaris.imag.fr/nicolas.gast/ – nicolas.gast@inria.fr

Questions?