Bandit algorithms for tree search
Applications to games, optimization, and planning

Rémi Munos

SequeL project: Sequential Learning
http://sequel.futurs.inria.fr/

INRIA Lille - Nord Europe
INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE
INRIA

Journées MAS de la SMAI, Rennes 27-29 Août 2008

Bandit algorithms for tree search
Applications to games, optimization, and planning

Outline of the talk:

- The multi-armed bandit problem
- A hierarchical of bandits
    - Application to tree search
    - Application to optimization
    - Application to planning

Exploration vs Exploitation in decision making

In an uncertain world, maybe partially observable, maybe adversarial, how should we make decisions?

- **Exploit**: act optimally according to our current beliefs
- **Explore**: learn more about the environment

Tradeoff between exploration and exploitation.
Appears in optimization/learning problems, such as in reinforcement learning.

Introduction to multi-armed bandits

General setting:

- At each round, several options (actions) are available to choose from.
- A reward is provided according to the choice made.
- Our goal is to optimize the sum of rewards.

Many potential applications:

- Clinical trials
- Advertising: what ad to put on a web-page?
- Labor markets: which job a worker should choose?
- Optimization of noisy function
- Numerical resource allocation

Example: a two-armed bandit

Say, there are 2 arms:


We have pulled the arms so far:

Time	1	2	3	4	5	6	7	8	...
Arm pulled	1	2	1	1	2	1	1	1	
Reward arm 1	10		9	11		12	8	10	
Reward arm 2		0			14				

Which arm should we pull next?

- What are the assumption about the rewards?
- What is really our goal?

The stochastic bandit problem

Setting:

- Set of K arms, defined by random variables $X_k \in [0,1]$, whose law is unknown,
- At each time t, choose an arm k_t and receive reward $x_t \overset{i.i.d.}{\sim} X_{k_t}$.

Goal: find an arm selection policy such as to maximize the expected sum of rewards.

Definitions :

- Let $\mu_k = \mathbb{E}[X_k]$ be the expected value of arm k.
- Let $\mu^* = \max_k \mu_k$ the optimal value, and k^* an optimal arm.

Exploration-exploitation tradeoff

Define the cumulative **regret**:

$$R_n \stackrel{\text{def}}{=} \sum_{t=1}^{n} \mu^* - \mu_{k_t}.$$

Property: Write $\Delta_k \stackrel{\text{def}}{=} \mu^* - \mu_k$, then

$$R_n = \sum_{k=1}^{K} n_k \Delta_k,$$

with n_k the number of times arm k has been pulled up to time n.
(regret results from pulling sub-optimal arms because of lack of information about an optimal one)

Goal: Find an arm selection policy such as to minimize R_n.

- Should we explore or exploit?
- Asymptotically consistent? (per-round regret $R_n/n \to 0$, i.e. $\frac{1}{n}\sum_t \mu_{k_t} \to \mu^*$).

Proposed solutions to the bandit problem?

This is an old problem! [Robbins, 1952]
(maybe surprisingly) not fully solved yet!
Many proposed solutions. Examples:

- ϵ**-greedy exploration**: choose apparent best action with proba $1-\epsilon$, or random action with proba ϵ,
- **Bayesian exploration**: assign prior to the arm distributions and based on the rewards, choose the arm with best posterior mean, or with highest probability of being the best
- **Optimistic exploration**: choose an arm that has a possibility of being the best
- **Boltzmann exploration**: choose arm k with proba $\propto \exp(\frac{1}{T}\widehat{X}_k)$
- etc.

The UCB algorithm

Upper Confidence Bounds algorithm [Auer et al. 2002]: at each time n, select an arm

$$\arg\max_k B_{k,n_k,n},$$

with

$$B_{k,n_k,n} \stackrel{\mathrm{def}}{=} \underbrace{\frac{1}{n_k}\sum_{s=1}^{n_k} x_{k,s}}_{\widehat{X}_{k,n_k}} + \underbrace{\sqrt{\frac{2\log(n)}{n_k}}}_{c_{n_k,n}},$$

where

- n_k is the number of times arm k has been pulled up to time n
- $x_{k,s}$ is the s-th reward obtained when pulling arm k.

Note that

- Sum of an exploitation term and an exploration term.
- $c_{n_k,n}$ is a confidence interval term, so $B_{k,n_k,n}$ is a UCB.

Intuition behind the UCB algorithm

Idea:

- Select an arm that has a high probability of being the best, given what has been observed so far.
- **"Optimism under the face of uncertainty"** strategy

Why?

- The B-values $B_{k,n_k,n}$ are Upper-Confidence-Bounds on μ_k: Indeed, from Chernoff-Hoeffding inequality,

$$\mathbb{P}\left(\widehat{X}_{k,t} + \sqrt{\frac{2\log(n)}{t}} \leq \mu_k\right) \leq e^{-2n\frac{2\log(n)}{t}} \leq n^{-4}.$$

Regret bound for UCB

Proposition

Each sub-optimal arm k is visited in average, at most:

$$\mathbb{E} n_k(n) \leq 8 \frac{\log n}{\Delta_k^2} + \ cst$$

times (where $\Delta_k \stackrel{\mathrm{def}}{=} \mu^* - \mu_k > 0$).

Thus the expected regret is bounded by:

$$\mathbb{E} R_n = \sum_k \mathbb{E}[n_k] \Delta_k \leq 8 \sum_{k:\Delta_k>0} \frac{\log n}{\Delta_k} + \ \text{cst.}$$

This is optimal (up to sub-log terms) since $\mathbb{E} R_n = \Omega(\log n)$ [Lai and Robbins, 1985].

Intuition of the proof

Let k be a sub-optimal arm, and k^* be an optimal arm. At time n, if arm k is selected, this means that

$$\begin{aligned}
B_{k,n_k,n} &\geq B_{k^*,n_{k^*},n} \\
\widehat{X}_{k,n_k} + \sqrt{\frac{2\log(n)}{n_k}} &\geq \widehat{X}_{k^*,n_{k^*}} + \sqrt{\frac{2\log(n)}{n_{k^*}}} \\
\mu_k + 2\sqrt{\frac{2\log(n)}{n_k}} &\geq \mu^*, \text{ with high proba} \\
n_k &\leq \frac{8\log(n)}{\Delta_k^2}
\end{aligned}$$

Thus with high probability, if $n_k > \frac{8\log(n)}{\Delta_k^2}$, then arm k will not be selected. Thus $n_k \leq \frac{8\log(n)}{\Delta_k^2} + 1$ with high proba.

Sketch of proof

Write $u = \frac{8\log(n)}{\Delta_k^2} + 1$. We have:

$$
\begin{aligned}
n_k(n) - u &\leq \sum_{t=u+1}^{n} \mathbf{1}_{k_t = k; n_k(t) > u} \leq \sum_{t=u+1}^{n} \mathbf{1}_{\exists s: u < s \leq t, \exists s^*: 1 \leq s^* \leq t, \text{ s.t. } B_{k,s,t} \geq B_{k^*,s^*,t}} \\
&\leq \sum_{t=u+1}^{n} \left[\mathbf{1}_{\exists s: u < s \leq t \text{ s.t. } B_{k,s,t} > \mu^*} + \mathbf{1}_{\exists s^*: 1 \leq s^* \leq t \text{ s.t. } B_{k^*,s^*,t} \leq \mu^*} \right] \\
&\leq \sum_{t=u+1}^{n} \left[\sum_{s=u+1}^{t} \mathbf{1}_{B_{k,s,t} > \mu^*} + \sum_{s=1}^{t} \mathbf{1}_{B_{k^*,s,t} \leq \mu^*} \right]
\end{aligned}
$$

Now, taking the expectation of both sides,

$$
\begin{aligned}
\mathbb{E}[n_k(n)] - u &\leq \sum_{t=u+1}^{n} \left[\sum_{s=u+1}^{t} \mathbb{P}\big(B_{k,s,t} > \mu^*\big) + \sum_{s=1}^{t} \mathbb{P}\big(B_{k^*,s,t} \leq \mu^*\big) \right] \\
&\leq \sum_{t=u+1}^{n} \left[\sum_{s=u+1}^{t} t^{-4} + \sum_{s=1}^{t} t^{-4} \right] \leq \frac{\pi^2}{3}
\end{aligned}
$$

PAC-UCB

Let $\beta > 0$, by slightly changing the confidence interval term, i.e.

$$B_{k,t} \stackrel{\text{def}}{=} \widehat{X}_{k,t} + \sqrt{\frac{\log(Kt^2\beta^{-1})}{t}},$$

then

$$\mathbb{P}\left(\left|\widehat{X}_{k,t} - \mu_k\right| \leq \sqrt{\frac{\log(Kt^2\beta^{-1})}{t}}, \forall k \in \{1, \dots, K\}, \forall t \geq 1\right) \geq 1 - \beta.$$

PAC-UCB [Audibert et al. 2007]: with probability $1 - \beta$, the regret is bounded by a constant independent of n:

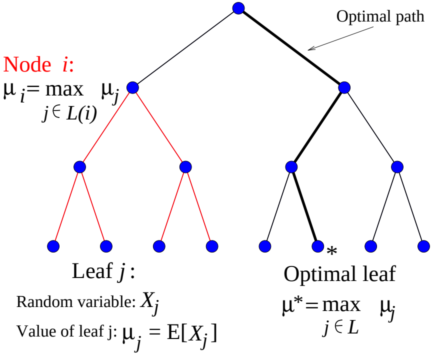
$$R_n \leq 6 \log(K\beta^{-1}) \sum_{k:\Delta_k > 0} \frac{1}{\Delta_k}.$$

Hierarchy of bandits

- Bandit (or regret minimization) algorithms = methods for rapidly selecting the best action.
- **Hierarchy of bandits**: the reward obtained when pulling an arm is itself the return of another bandit in a hierarchy. Applications to
    - tree search,
    - optimization,
    - planning

The tree search problem

- To each leaf $j \in \mathcal{L}$ of a tree is assigned a random variable $X_j \subset [0,1]$ whose law is unknown.
- At each time t, a leaf $I_t \in \mathcal{L}$ is selected and a reward $x_t \overset{iid}{\sim} X_{I_t}$ is received.

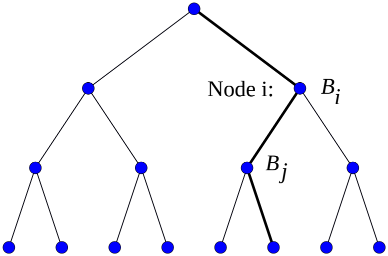


Goal: find an exploration policy that maximizes the expected sum of obtained rewards.

Idea: use bandit algorithms for efficient tree exploration

UCB-based leaf selection policy

Leaf selection policy:
To each node i is assigned a value B_i.
The chosen leaf I_t is selected by following a path from the root to a leaf, where at each node i, the next node (child) is the one with highest B-value.



Goal: Design B-values (upper bounds on the true values μ_i of each node i) such that the resulting leaf selection policy maximizes the expected sum of obtained rewards.

Flat UCB

We implement UCB directly on the leaves:

$$B_i \stackrel{\text{def}}{=} \begin{cases} \widehat{X}_{i,n_i} + \sqrt{\frac{2\log(n_p)}{n_i}} & \text{if } i \text{ is a leaf,} \\ \max_{j\in\mathcal{C}(i)} B_j & \text{otherwise.} \end{cases}$$

Property (Chernoff-Hoeffding): With high probability, we have $B_i \geq \mu_i$, for all nodes i.
Bound on the regret: any sub-optimal leaf j is visited in expectation at most $\mathbb{E}n_j = O(\log(n)/\Delta_j^2)$ times (where $\Delta_j = \mu^* - \mu_j$). Thus, the regret is bounded by:

$$\mathbb{E}R_n = O\Big(\log(n) \sum_{j\in\mathcal{L},\mu_j<\mu^*} \frac{1}{\Delta_j}\Big).$$

Problem: all leaves must be visited at least once!

UCT (UCB applied to Trees)

UCT [Kocsis and Szepesvári, 2006]:

$$B_i \stackrel{\text{def}}{=} \widehat{X}_{i,n_i} + \sqrt{\frac{2\log(n_p)}{n_i}}.$$

Intuition:

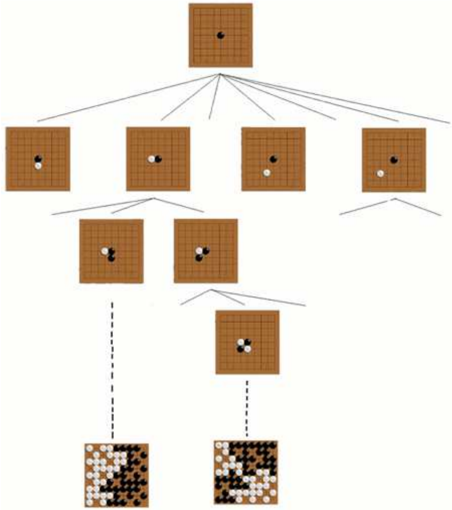
- Explore first the most promising branches
- Adapts automatically to the effective smoothness of the tree

Very good results in computer-go

The MoGo program

Collaborative work with Yizao Wang, Sylvain Gelly, Olivier Teytaud and many others. See [Gelly et al., 2006].

- Explore-Exploit with UCT (Min-Max)
- Monte-Carlo evaluation
- Asymmetric tree expansion
- Anytime algo
- Use of features
- World computer-go champion



Interestingly: stochastic methods for deterministic problem!

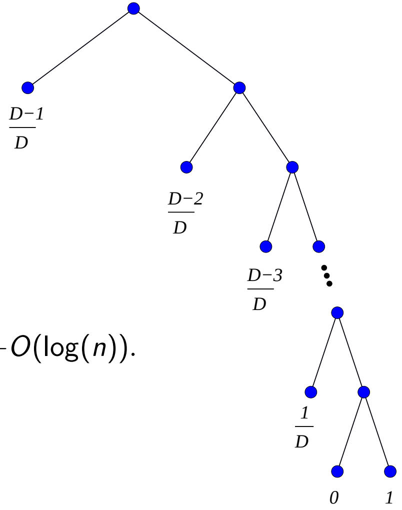
Analysis of UCT

Properties:

- The obtained rewards at a (non-leaf) node i are not i.i.d.
- Thus the B values are not upper confidence bounds on the node values
- However, all leaves are eventually visited infinitely,
- thus the algorithm is eventually consistent: the regret is $O(log(n))$ after an initial period...
- which may last very ... very long!

Bad case for UCT

Consider the tree:



The left branches seem to be the best thus are explored for a **very** long time before the optimal leaf is eventually reached.
The expected regret is disastrous:

$$\mathbb{E}R_n = \Omega(\underbrace{\exp(\exp(\ldots\exp(}_{D \text{ times}}1)\ldots)))+O(\log(n)).$$

Much much worst than uniform exploration!

In short...

So far we have seen:

- Flat-UCB: does not exploit possible smoothness, but very good in the worst case!
- UCT:
    - indeed adapts automatically to the effective smoothness of the tree,
    - but the price of this adaptivity may be very very high.
    - In good cases, UCT is VERY efficient!
    - In bad cases, UCT is VERY poor!

We should use the actual smoothness of the problem, if any, to design relevant algorithms.

BAST (Bandit Algorithm for Smooth Trees)

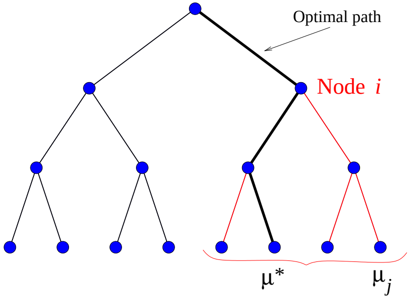
(Joint work with Pierre-Arnaud Coquelin)

Assumption: along an optimal path, for each node i of depth d, for all leaves $j \in \mathcal{L}(i)$,

$$\mu^* - \mu_j \le \delta_d,$$

where δ_d is a *smoothness function*

Examples: holds for function optimization or discounted control.



Define the B-values:

$$B_i \stackrel{\text{def}}{=} \min \begin{cases} \max_{j \in \mathcal{C}(i)} B_j, \\ \widehat{X}_{i,n_i} + \sqrt{\frac{2 \log(n_p)}{n_i}} + \delta_d \end{cases}$$

Remark:

UCT = (BAST with $\delta_d = 0$). Flat-UCB = (BAST with $\delta_d = \infty$).

Properties of BAST

Properties:

- These B-values are true upper confidence bounds on the optimal nodes value,
- The tree grows in an asymmetric way, leaving mainly unexplored the sub-optimal branches,
- Only the optimal path is essentially explored.

Regret analysis of BAST... will come in a moment as a special case of a more general framework (bandits in metric spaces).

Multi-armed bandits in metric spaces

Let X be a metric space with $l(x, y)$ a distance. Let $f(x)$ be a Lipschitz function:

$$|f(x) - f(y)| \leq l(x, y).$$

Write $f^* \stackrel{\text{def}}{=} \sup_{x \in X} f(x)$.
Multi-armed bandit problem on X: At each round t, choose a point (arm) x_t, receive reward r_t independent sample drawn from a distribution $\nu(x_t)$ with mean $f(x_t)$.
Goal: minimize regret: $R_n \stackrel{\text{def}}{=} \sum_{t=1}^{n} f^* - r_t$.
Examples:

- Tree search with smooth rewards
- Optimization in continuous space of a Lipschitz function, given noisy evaluations

Hierarchical Optimistic Optimization

(Joint work with S. Bubeck, G. Stoltz, Cs. Szepesvári)

- Consider a tree of partitions of X,
- Each node i corresponds to a domain D_i of the state space.

Write $diam(i) = \sup_{x,y \in D_i} l(x,y)$ the diameter of D_i. Let $\mathcal{T}_t$ denote the set of expanded nodes at round t.

Algorithm:

- Start with $\mathcal{T}_1 = \{\text{root}\}$. (whole domain X)
- At each round t, follow a path from the root to a leaf i_t of $\mathcal{T}_t$ by maximizing the B-values,
- Expand the node i_t: choose (arbitrarily) a point $x_t \in D_{i_t}$, and add i_t to $\mathcal{T}_t$,
- Observe reward $r_t \sim \nu(x_t)$ and update the B-values:

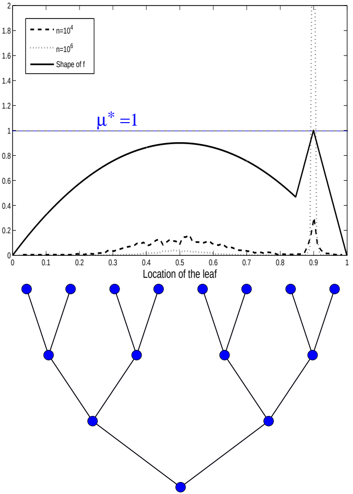
$$B_i \stackrel{\text{def}}{=} \min\Big[\max_{j \in \mathcal{C}(i)} B_j, \widehat{X}_{i,n_i} + \sqrt{\frac{2\log(n)}{n_i}} + diam(i)\Big],$$

Application to continuous optimization

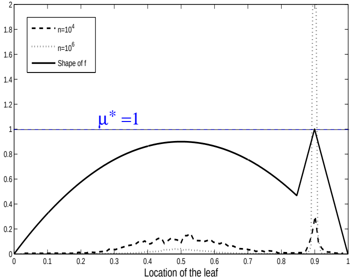
Problem:
Optimize a Lipschitz function f, given noisy evaluations.

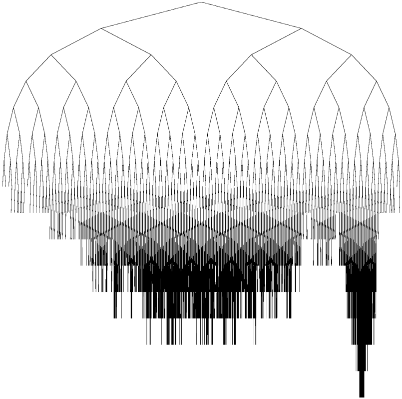
Example in 1d:
The (infinite) tree represents a binary splitting of $[0,1]$ at all scales.

Rewards:
$r_t \sim \mathcal{B}(f(x_t))$ a Bernoulli with parameter $f(x_t)$, where x_t is the chosen point at time t.
If f is L-Lipschitz, then the smoothness assumption holds with the metric $l(x,y) = L|x-y|$.



Resulting tree for the optimization problem





Resulting tree at stage $n = 4000$.

Analysis of the regret

- Let d be the **dimension** of X (ie. such that we need $O(\varepsilon^{-d})$ balls of radius ε to cover X). Then

$$\mathbb{E}R_n = O(n^{\frac{d+1}{d+2}}).$$

- We also have a lower bound $\mathbb{E}R_n = \Omega(n^{\frac{d+1}{d+2}})$ [Kleinberg et al., 2008]
- Let d' be the **near-optimality dimension** of f in X: i.e. such that we need $O(\varepsilon^{-d'})$ balls of radius ε to cover

$$X_\varepsilon \stackrel{\text{def}}{=} \{x \in X, f(x) \geq f^* - \varepsilon\}.$$

Then

$$\mathbb{E}R_n = O(n^{\frac{d'+1}{d'+2}}).$$

Much better!!!

Powerful generalization

Actually we don't need the assumption that X is metric, neither that f is Lipschitz! But (almost) only that f is *one-sided locally Lipschitz* around its max w.r.t. a *dissimilarity measure* l, i.e.

$$f^* - f(y) \leq l(x^*, y).$$

Interesting example:
Consider $X = [0,1]^d$. Assume that f is locally Hölder (with order α) around its maximum, i.e. $f^* - f(y) = \Theta(||x^* - y||^\alpha)$.
Then we may choose $l(x,y) = ||x-y||^\alpha$, and X_ε is is thus covered by $O(1)$ ball of radius ε. Thus the near-optimality dimension $d' = 0$, and the regret is:

$$\mathbb{E}R_n = O(\sqrt{n}),$$

whatever the dimension of the space d!
$\rightarrow$ **Optimization is not more difficult than integration**

Let's go back to the trees...

- but in a very simplified setting: rewards are deterministic
- Still we want to investigate the "optimistic" exploration strategy
- Application to planning

Application to planning

(Joint work with Jean-François Hren)

Consider a controlled **deterministic system with discounted rewards**.

- From the current state x_t, consider the look-ahead tree of all possible reachable states.
- Use n computational resources (CPU time, number of calls to a generative model) to explore the tree and return a proposed actions a_t
- This induces a policy π_n
- Goal: Minimize the loss resulting from using policy π_n instead of an optimal one:

$$R_n \stackrel{\mathrm{def}}{=} V^* - V^{\pi_n}$$

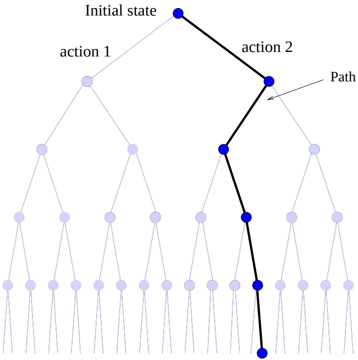
Look-ahead tree for planning in deterministic systems

At time t, for the current state x_t. Build the look-ahead tree:

- Root of the tree = current state x_t
- Nodes = reachable states by a sequence of actions,
- Receive discounted sum of rewards along the path:

$$\sum_{t\geq 0} \gamma^t r_t,$$

- Explore the tree using n computational resources
- Propose an action as close as possible to the optimal one



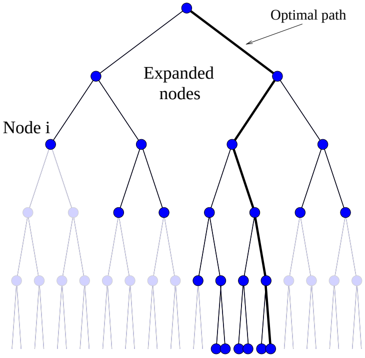
Optimistic exploration

(BAST/HOO algo in deterministic setting)

- For any node i of depth d, define the B-values:

$$B_i \stackrel{\text{def}}{=} \sum_{t=0}^{d-1} \gamma^t r_t + \frac{\gamma^d}{1-\gamma} \geq v_i$$

- At each round n, expand the node with highest B-value
- Observe reward, update B-values,
- Repeat until no more available resources
- Return maximizing action



Analysis of the regret

Define β such that the proportion of ϵ-optimal paths is $O(\epsilon^\beta)$. Let

$$\kappa \stackrel{\text{def}}{=} K\gamma^\beta \in [1, K].$$

- If $\kappa > 1$, then

$$R_n = O\left(n^{-\frac{\log 1/\gamma}{\log \kappa}}\right).$$

  (recall that for the uniform planning $R_n = O\big(n^{-\frac{\log 1/\gamma}{\log K}}\big)$.)

- If $\kappa = 1$, then $R_n = O\big(\gamma^{\frac{(1-\gamma)^\beta}{c} n}\big)$, where c defined by the proportion of ϵ-path being bounded by $c\epsilon^\beta$. This provides exponential rates.

Some intuition

Write $\mathcal{T}_\infty$ the tree of all expandable nodes:

$$\mathcal{T}_\infty = \{\text{node } i \text{ of depth } d \text{ s.t. } v_i + \frac{\gamma^d}{1-\gamma} \geq v^*\}$$

- $\mathcal{T}_\infty$ = set of nodes from which one cannot decide whether the node is optimal or not,
- At any round n, the set of expanded nodes $\mathcal{T}_n \subset \mathcal{T}_\infty$,
- κ = branching factor of $\mathcal{T}_\infty$.

The regret

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

comes from a search in the tree $\mathcal{T}_\infty$ with branching factor $\kappa \in [1, K]$.

Upper and lower bounds

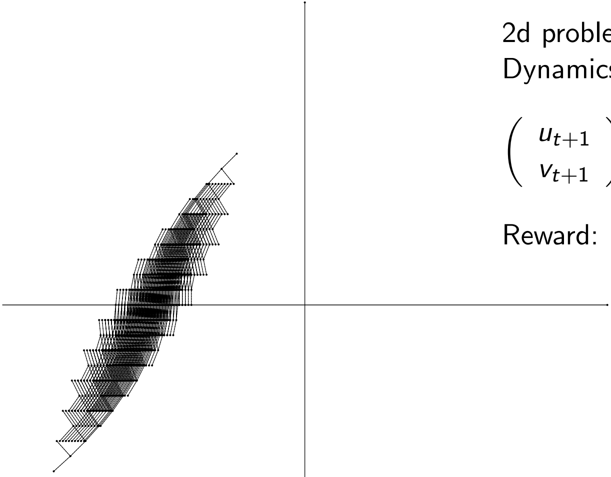
For any $\kappa \in [1, K]$.

- Define $\mathcal{P}_\kappa$ as the set of problems having a κ-value.
- For any problem $P \in \mathcal{P}_\kappa$, write $R_{\mathcal{A}(P)}(n)$ the regret of using the algorithm $\mathcal{A}$ on the problem P with n computational resources.

Then:

$$
\begin{aligned}
\sup_{P \in \mathcal{P}_\kappa} R_{\mathcal{A}_{uniform}(P)}(n) &= \Theta\left(n^{-\frac{\log 1/\gamma}{\log K}}\right) \\
\sup_{P \in \mathcal{P}_\kappa} R_{\mathcal{A}_{optimistic}(P)}(n) &= \Theta\left(n^{-\frac{\log 1/\gamma}{\log \kappa}}\right).
\end{aligned}
$$

Numerical illustration



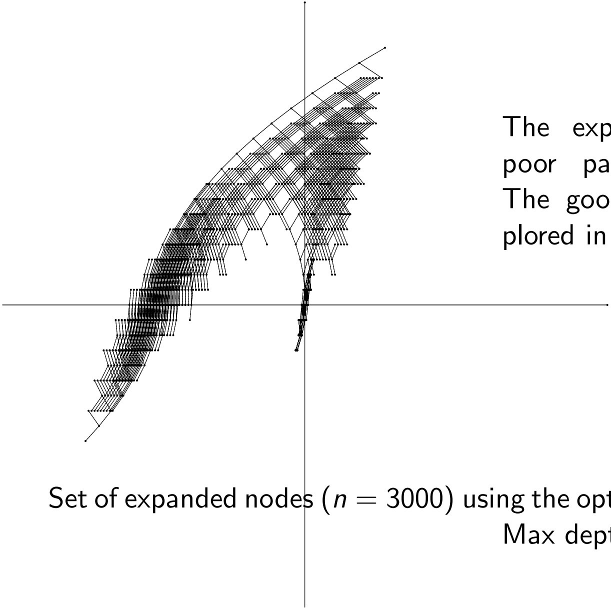
2d problem: $x = (u, v)$.
Dynamics:

$$\begin{pmatrix} u_{t+1} \\ v_{t+1} \end{pmatrix} = \begin{pmatrix} u_t + v_t \Delta_t \\ v_t + a_t \Delta t \end{pmatrix}$$

Reward: $r(u, v) = -u^2$.

Set of expanded nodes ($n = 3000$) using the uniform planning.
Max depth $= 10$.

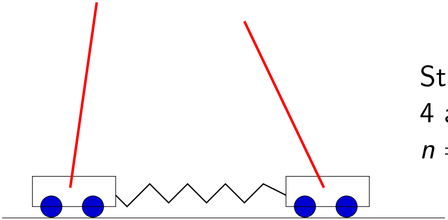
Numerical illustration



The exploration of the poor paths is shallow. The good paths are explored in deeper depths.

Set of expanded nodes ($n = 3000$) using the optimistic planning.
Max depth $= 43$.

Two inverted pendulum linked with a spring



State space of dimension 8

4 actions

$n = 3000$ at each iteration

References

- J.Y. Audibert, R. Munos, and C. Szepesvari, *Tuning bandit algorithms in stochastic environments*, ALT, 2007.
- P. Auer, N. Cesa-Bianchi, and P. Fischer, *Finite time analysis of the multiarmed bandit problem*, Machine Learning, 2002.
- S. Bubeck, R. Munos, G. Stoltz, Cs. Szepesvari *Online Optimization in X-armed Bandits*, submitted to NIPS, 2008.
- P.-A. Coquelin and R. Munos, *Bandit Algorithm for Tree Search*, UAI 2007.
- S. Gelly, Y. Wang, R. Munos, and O. Teytaud, *Modification of UCT with Patterns in Monte-Carlo Go*, RR INRIA, 2006.

References (cont'ed)

- J.-F. Hren and R. Munos, *Optimistic planning in deterministic systems*. Research report INRIA, 2008.
- M. Kearns, Y. Mansour, A. Ng, *A Sparse Sampling Algorithm for Near-Optimal Planning in Large Markov Decision Processes*, Machine Learning, 2002.
- R. Kleinberg, *Nearly tight bounds for the continuum-armed bandit problem*, NIPS 2004.
- R. Kleinberg, A. Slivkins, and E. Upfal, *Multi-Armed Bandits in Metric Spaces*, ACM Symposium on Theory of Computing, 2008.
- L. Kocsis and Cs. Szepesvári, *Bandit based Monte-Carlo Planning*, ECML 2006.
- T. L. Lai and H. Robbins, *Asymptotically Efficient Adaptive Allocation Rules*, Advances in Applied Mathematics, 1985.