Online optimistic planning for Markov decision processes

Lucian Buşoniu

ACAI SSRL, Nieuwpoort, 10 October 2017

Part I

Introduction. Deterministic case

Model-based motivation

In practice, a model may be available
(sometimes precise, sometimes rough)

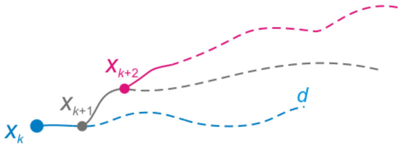
$\Rightarrow$ **Use it!**

Model-based techniques still very useful due to generality
(nonlinear, stochastic problems)

Online planning idea

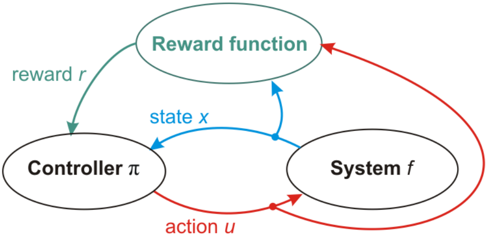
At each step, use model to solve problem locally:

1. Explore action sequences from current state, to find a near-optimal sequence
2. Apply first action of this sequence, and repeat



Receding-horizon model-predictive control

Deterministic MDP: Control perspective

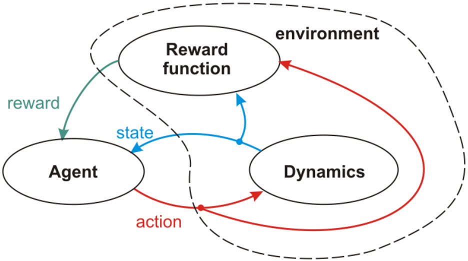


- At step k, controller measures states x, applies actions u
- System: dynamics $x_{k+1} = f(x_k, u_k)$
- Performance: reward function $r_{k+1} = \rho(x_k, u_k)$
- **Objective**: find policy $u = \pi(x)$ that maximizes return

$$\sum_{k=0}^{\infty} \gamma^k r_{k+1}$$

with discount factor $\gamma \in (0, 1)$

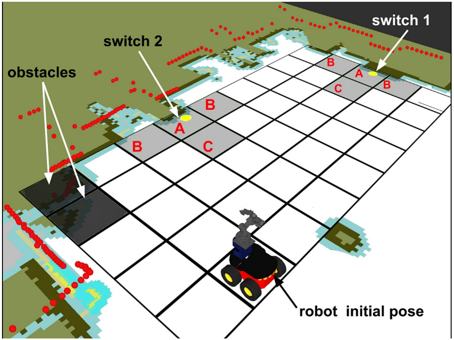
AI perspective



- Agent observes state, applies action
- Environment changes state according to dynamics

  ... and sends back a reward, according to reward function
- **Objective:** maximize discounted return

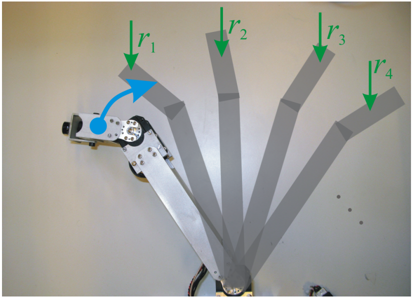
Example: Domestic robot



A domestic robot ensures light switches are off
Abstractization to high-level control (physical actions implemented by low-level controllers)

- States: grid coordinates, switch states
- Actions: movements NSEW, toggling switch
- Rewards: when switches toggled on→off

Example: Robot arm

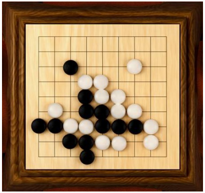


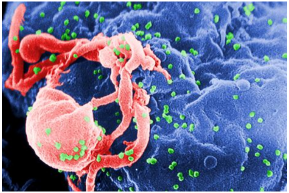
Low-level control

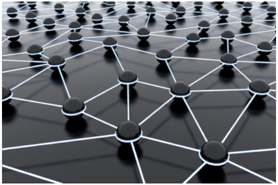
- States: link angles and angular velocities
- Actions: motor voltages
- Rewards: e.g. to reach a desired configuration, give larger rewards as robot gets closer to it

Many other applications

Artificial intelligence, medicine, multiagent systems, economics etc.







Value function and optimal solution

- V-function of policy π:

$$V^\pi(x) = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, \pi(x_k))$$

  where $x_0 = x, x_{k+1} = f(x_k, \pi(x_k))$
- Optimal V-function: $V^*(x) = \max_\pi V^\pi(x)$
- Bellman equation for V^π:

$$V^\pi(x) = \rho(x, \pi(x)) + \gamma V^\pi(f(x, \pi(x)))$$

- Bellman optimality equation (for V^*):

$$V^*(x) = \max_u [\rho(x, u) + \gamma V^*(f(x, u))]$$

- Once V^* available, optimal policy is:

$$\pi^*(x) = \arg\max_u [\rho(x, u) + \gamma V^*(f(x, u))]$$

Value iteration

Turn Bellman optimality equation:

$$V^*(x) = \max_u [\rho(x, u) + \gamma V^*(f(x, u))]$$

into an iterative assignment:

Value iteration

repeat at each iteration t
  for all x **do**
    $V_{t+1}(x) = \max_u [\rho(x, u) + \gamma V_t(f(x, u))]$
  end for
until convergence to V^*
$\pi^*(x) = \arg\max_u [\rho(x, u) + \gamma V^*(f(x, u))]$

Monotonic, exponential convergence

Lecture structure

Online, optimistic planning in:

1. Deterministic MDPs
2. Stochastic MDPs and adversarial problems
3. Continuous-action MDPs (+ final remarks)

Practical session: Implement & try deterministic planner

1. Idea & background
2. Optimistic planning for deterministic systems
3. Analysis
4. Example and real-time application
5. Relation to value-function methods

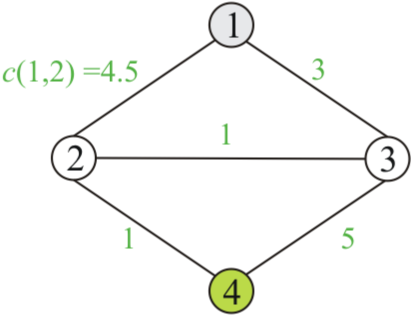
Relation to classical planning

Most methods we discuss are **extensions of classical planning** (A*, AO*, B*) to solving MDPs

We provide **near-optimality guarantees** as a function of computation n and of complexity κ of the problem:

$$\text{error} = O(g(n, \kappa))$$

Shortest-path graph search



- Graph with costs $c(i,j)$ for traveling between nodes i and j
- **Objective:** lowest-cost path from start s to target t (1 to 4)

Classical A*

Uses a heuristic $\delta(i) \leq$ the lowest cost from i to the target t

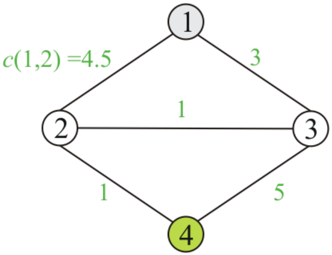
A* (tree-search version)

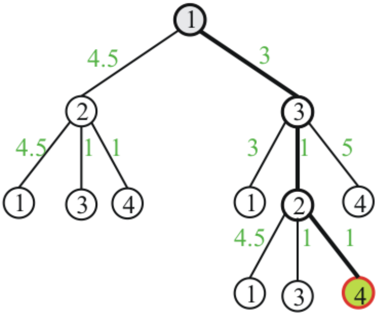
initialize tree with start node s, set $\ell(s) = 0, b(s) = \delta(s)$
loop
  select leaf $i^\dagger$ with lowest b
  if $i^\dagger =$ target t, stop
  expand $i^\dagger$ with all neighbors j
  for each j, $\ell(j) = \ell(i) + c(i, j)$, $b(j) = \ell(j) + \delta(j)$
end loop
return path from s to t

Each node evaluated by underestimate b of the lowest-cost path going through it – **optimism under uncertainty**

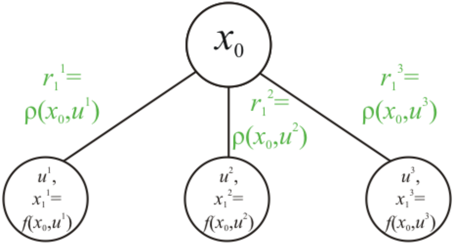
A* on example graph

Take $\delta(i) = 1$, smallest possible cost



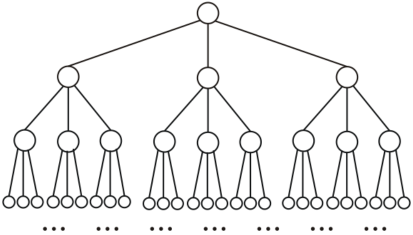


Applying A* idea to MDPs



- Each tree node gets the meaning of state
- One child for each action, each transition associated with a reward (instead of cost)

Applying A* idea to MDPs (cont'd)



- Problem is infinite-horizon, tree is infinitely deep
- Optimal solution also infinitely deep in general
  $\Rightarrow$ must stop suboptimally
- Suboptimal solution finite in length
  $\Rightarrow$ work in receding horizon
- Maximize discounted returns instead of minimizing costs
  $\Rightarrow$ optimistic value should **over**estimate return

Formal setting

Assumptions

- Finite, discrete action space $U = \{u^1, \dots, u^M\}$
- Bounded reward function $\rho(x, u) \in [0, 1], \forall x, u$

Denote current step by 0 (by convention). Then:

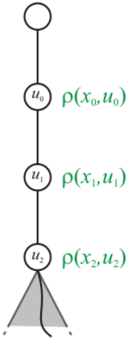
- Infinite action sequences: $\boldsymbol{u}_\infty = (u_0, u_1, \dots)$
- Solve $\sup_{\boldsymbol{u}_\infty} v(\boldsymbol{u}_\infty) := \sum_{k=0}^{\infty} \gamma^k r_{k+1}$

Formal setting: Values

- Finite sequence $\boldsymbol{u}_d$ also seen as **set** of infinite sequences $(u_0, \dots, u_{d-1}, \star, \star, \dots)$

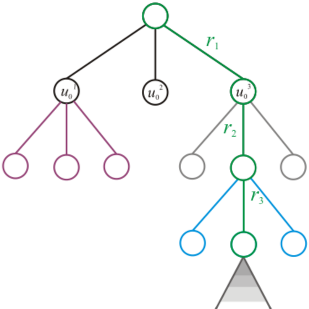
- $\ell(\boldsymbol{u}_d) = \sum_{k=0}^{d-1} \gamma^k \rho(x_k, u_k)$
lower bound on returns of $\boldsymbol{u}_\infty \in \boldsymbol{u}_d$
- $b(\boldsymbol{u}_d) = \ell(\boldsymbol{u}_d) + \frac{\gamma^d}{1-\gamma} =: \delta(d)$, **diameter**
optimistic upper bound on the returns

- $v(\boldsymbol{u}_d) = \sup_{\boldsymbol{u}_\infty \in \boldsymbol{u}_d} v(\boldsymbol{u}_\infty)$
value of applying $\boldsymbol{u}_d$ and then acting optimally



Optimistic planning for deterministic systems (OPD)

initialize empty sequence $\boldsymbol{u}_0$ (= all infinite sequences)
for $t = 1$ to n **do**
  select **optimistic** leaf sequence $\boldsymbol{u}_t^\dagger$, maximizing b
  expand $\boldsymbol{u}_t^\dagger$: children for all actions, setting ℓ and b
end for
return greedy $\boldsymbol{u}_{d*}^*$ maximizing ℓ



(Hren & Munos, 2008)

Relation to bandit problems

Besides obvious relation with RL (we solve the problem model-based), there is a deeper connection via **exploration**

At single state, exploration modeled as **multi-armed bandit**:

- Action j = arm with reward distribution ρ_j, expectation μ_j
- Best arm (optimal action) has expected value μ^*
- At step k, we pull arm (try action) j_k, getting $r_k \sim \rho_{j_k}$
- **Objective:** After n pulls, small regret: $\sum_{k=1}^{n} \mu^* - \mu_{j_k}$

Relation to bandit problems (cont'd)

Good idea: after n steps, pick arm with largest **upper confidence bound**:

$$b(j) = \hat{\mu}_j + \sqrt{\frac{3 \log n}{2 n_j}}$$

where:

- $\hat{\mu}_j$ = mean of rewards observed for arm j so far
- n_j how many times arm j was pulled

Optimism in the face of uncertainty

- Bandits: uncertainty = unknown reward distributions
- Planning: uncertainty = incomplete (finite-horizon) solutions

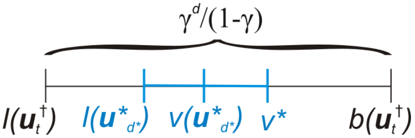
Near-optimality vs. depth

1. OPD returns a sequence $\boldsymbol{u}^*_{d^*}$, with length $d^* =$ the deepest expanded d
2. This sequence is near-optimal up to deepest diameter:

$$v^* - v(\boldsymbol{u}^*_{d^*}) \leq \delta(d^*) = \frac{\gamma^{d^*}}{1-\gamma}$$

where v^* the optimal value (at x_0)

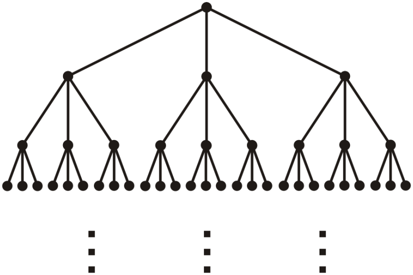
Near-optimality proof



- For any iteration t, $b(\boldsymbol{u}_t^\dagger) \geq v^*$ since it's larger than the b-value of any leaf (including that on the optimal path)
- At the end, $\ell(\boldsymbol{u}_{d^*}^*)$ is larger than any ℓ-value, in particular than $\ell(\boldsymbol{u}_t^\dagger)$
- But the gap $b(\boldsymbol{u}_t^\dagger) - \ell(\boldsymbol{u}_t^\dagger) = \frac{\gamma^d}{1-\gamma}$ with d the depth of $\boldsymbol{u}_t^\dagger$!
  This holds e.g. at d^*
- Finally, $v(\boldsymbol{u}_{d^*}^*) \geq \ell(\boldsymbol{u}_{d^*}^*)$

Case 1: All paths optimal

Take a tree where all rewards are 1:



$b(\boldsymbol{u}_d) = \frac{1}{1-\gamma}, \ \forall \boldsymbol{u}_d \ \Rightarrow$ OPD expands uniformly, breadth-first

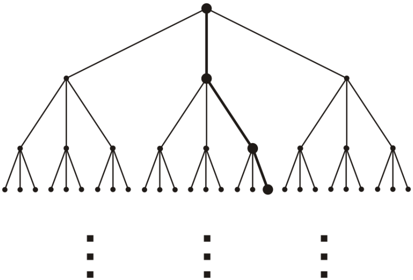
So to expand all nodes down to depth d, we must spend:

$$n = \sum_{i=0}^{d} M^i = \frac{M^{d+1} - 1}{M - 1}$$

and the tree grows very slowly with budget n

Case 2: One path optimal

Take a tree where rewards are 1 only along a single path (thick line), and 0 everywhere else:



$b(\boldsymbol{u}_d) = \frac{1}{1-\gamma}$ only on optimal path, $\frac{\gamma^d}{1-\gamma}$ elsewhere
$\Rightarrow$ OPD expands only the optimal path

So to expand down to depth d, we must spend only $n = d$, and the tree grows very fast with n

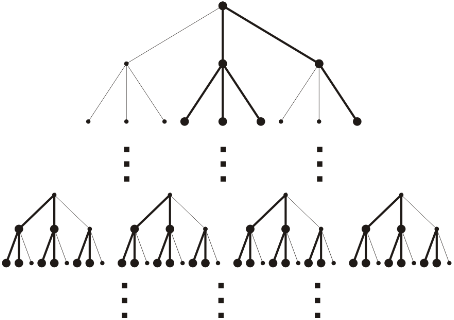
General case: Branching factor

- Algorithm only expands in near-optimal subtree:

$$\mathcal{T}^* = \{\boldsymbol{u}_d \mid v^* - v(\boldsymbol{u}_d) \le \delta(d)\}$$

- Define κ = asymptotic branching factor of $\mathcal{T}^*$:
  problem complexity measure, $\kappa \in [1, M]$
  (related to effective branching factor of A*)

E.g. $\kappa = 2$, $M = 3$:



Depth vs. budget n

To reach depth d in tree with branching factor κ,
we must expand $n = \mathrm{O}(\kappa^d)$ nodes

$$\Rightarrow \quad d^* = \Omega\left(\frac{\log n}{\log \kappa}\right)$$

Final guarantee: Near-optimality vs. budget

Theorem

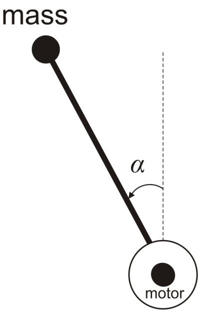
- OPD returns a long sequence $\boldsymbol{u}^*_{d^*}$, $d^* = \Omega(\frac{\log n}{\log \kappa})$
- This sequence is near-optimal:

$$v^* - v(\boldsymbol{u}^*_{d^*}) \leq \delta(d^*) = \frac{\gamma^{d^*}}{1-\gamma} = \begin{cases} \mathrm{O}(n^{-\frac{\log 1/\gamma}{\log \kappa}}) & \text{if } \kappa > 1 \\ \mathrm{O}(\gamma^{n/C}) & \text{if } \kappa = 1 \end{cases}$$

- Generality paid by exponential computation $n = \mathrm{O}(\kappa^d)$
- But κ can be small in interesting problems!

(Hren & Munos, 2008)

Example: Inverted pendulum



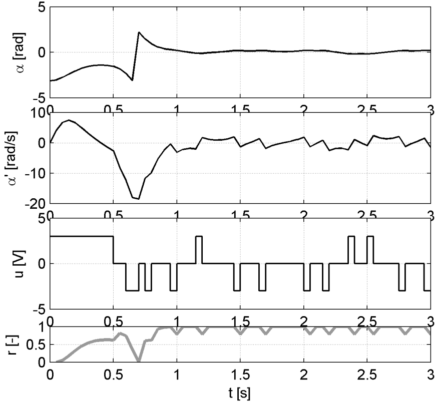
- $x = [\text{angle } \alpha, \text{ velocity } \dot{\alpha}]^\top$
- u = voltage
- $\rho(x, u) = -x^\top Q x - u^\top R u$
- Discount factor $\gamma = 0.98$

- Objective: stabilize pointing up
- Insufficient torque $\Rightarrow$ swing-up required

Simulation: Inverted pendulum

Demo

Swingup trajectory:

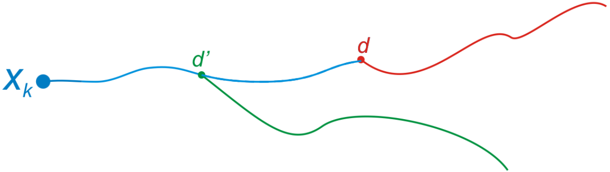


Real-time idea

Challenge: computation time large and must be handled!

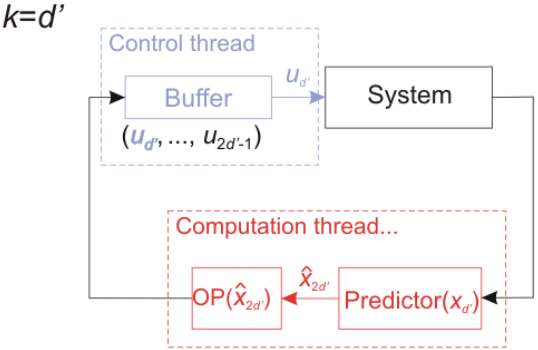
- Usually only first action of each sequence is sent to actuator
- But remember: OPD returns **long sequences**!

$\Rightarrow$ Send a longer subsequence (length d'),
and **use the time to compute in the background**



Real-time architecture

- Compute initial sequence (system assumed stable)
- Send to buffer, and immediately start computing next sequence from predicted state



(CSCS 2015)

Setting up real-time OPD

- We usually want to use all available time: $n = \left\lfloor d' \frac{T_s}{T_e} \right\rfloor$.

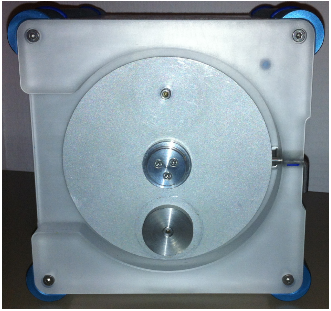
$\Rightarrow$ Select subsequence length d' so that:

$$d' \frac{T_s}{T_e} - \kappa^{d'/c} - 1 \geq 0$$

- Or, when κ, c unknown:

$$(d' \frac{T_s}{T_e} - 1)(K - 1) - K^{d'+1} + 1 \geq 0$$

Real-time results: Inverted pendulum

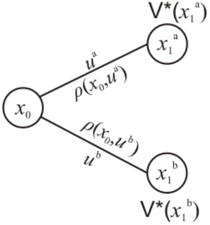


Relation to VI: 1 step

V^* available: search just one step ahead:

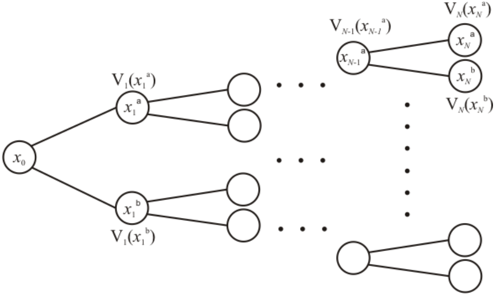
$$u_0 = \pi^*(x_0) = \arg\max_u [\rho(x_0, u) + \gamma V^*(f(x_0, u))]$$
$$= \arg\max_u [\rho(x_0, u) + \gamma V^*(x_1)]$$

Equivalent to a simple tree:



Relation to VI: N steps

V^* unavailable: search N steps ahead, $N \gg$:

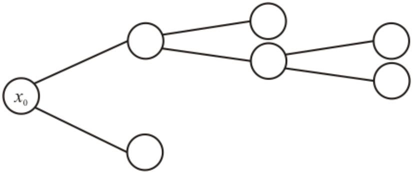


Equivalent to **local V-iteration** (backward view):

$V_N(x_N) \leftarrow 0$, for states x_N reachable from x_0
for $i = N-1, N-2, \ldots, 1$ **do**
  $V_i(x_i) = \max_u [\rho(x_i, u) + \gamma V_{i+1}(f(x_i, u))]$, $\forall x_i$ reachable
end for
$u_0 = \arg\max_u [\rho(x_0, u) + \gamma V_1(f(x_0, u))]$

Relation to VI: OPD tree

OPD actually explores the tree optimally:



and local VI works for this tree as well:

$V(x) \leftarrow 0$, for terminal nodes
for internal nodes, back to the root **do**
$\quad V(x) = \max_u [\rho(x, u) + \gamma V(f(x, u))]$
end for
$u_0 = \arg\max_u [\rho(x_0, u) + \gamma V(f(x_0, u))]$

Relation to VI

- VI gives global solution, OPD just local at x_0
- OPD insensitive to the complexity of the state space, which highly influences VI
- OPD complexity grows fast with number of actions
  $\Rightarrow$ appropriate for small actions spaces

Using V-functions in OPD

Instead of uninformed upper bounds $\gamma^d \frac{1}{1-\gamma}$,
use **good V-function estimates** at the leaves:

$$\gamma^d \widehat{V}(x_d)$$

- Similar to informed heuristics in A*
- Estimates could come from: rough initial value or policy iteration, online learning, etc.
- As long as $\widehat{V}(x) \geq V^*(x)$, algorithm improves (just like A*)
- Even if $\widehat{V}$ underestimates V^* by ε, algorithm improved when ε is small

(EAAI 2016)

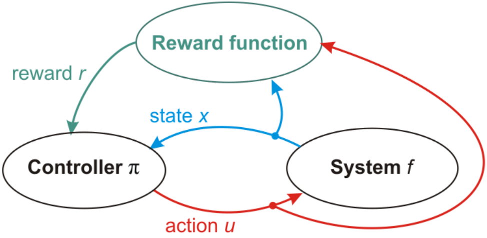
References for Part I

- **Textbook:** Munos, *From Bandits to Monte Carlo Tree Search: The Optimistic Principle Applied to Optimization and Planning*, Foundations and Trends in Machine Learning, 7, 2014.
- Hren, Munos, *OP of deterministic systems*, EWRL 2008.
- Wensveen, Busoniu, Babuska, *Real-Time Optimistic Planning with Action Sequences*, CSCS 2015.
- Busoniu, Daniels, Babuska, *Online Learning for Optimistic Planning*, Engineering Applications of AI, 2016.

Part II

Stochastic and adversarial problems

Stochastic case



In response to u in x, system no longer reacts deterministically – it can reach one of several states with different probabilities

Stochastic MDP and objective

Stochastic MDP

1. State and action spaces X, U keep their meaning
2. Transition function gives probabilities $\tilde{f}(x, u, x')$, $\tilde{f} : X \times U \times X \to [0, 1]$
3. Reward a function of the whole transition $\tilde{\rho}(x, u, x')$, $\tilde{\rho} : X \times U \times X \to \mathbb{R}$

Objective: find policy π to maximize expected return:

$$R^\pi(x_0) = \mathrm{E}\left\{ \sum_{k=0}^{\infty} \gamma^k \tilde{\rho}(x_k, \pi(x_k), x_{k+1}) \right\}$$

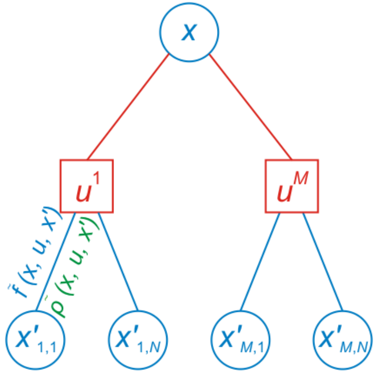
from any x_0

Formal setting

Assumptions

1. Finite, discrete action space $U = \{u^1, \dots, u^M\}$
2. Each action leads to (at most) N different next states
3. Bounded rewards $r \in [0, 1]$

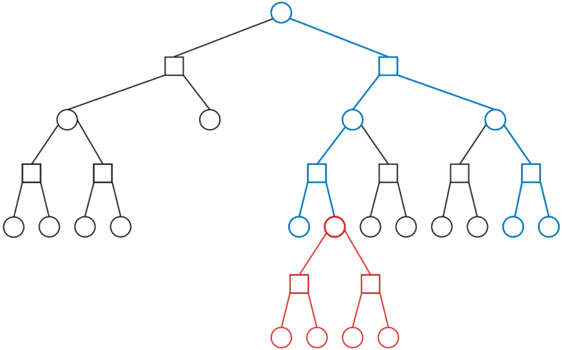
Tree structure



- Each of the M actions gets its separate node and has its N possible next states as children
- Probability and reward labels on action-next state arcs

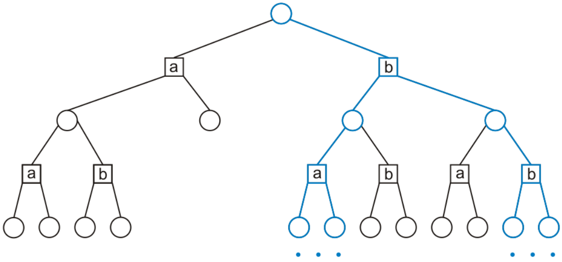
Algorithm outline

- Build tree by iteratively expanding state nodes (adding all M action children and $N \cdot M$ state children)
- Each expansion: select optimistic partial solution and its most useful leaf



Solution concept

- Closed-loop planning policy h:
  assigns action choices to all possible outcomes
- Value $v(h)$ = expected return while following h
- Restriction to finite subtree $\Rightarrow$ policy set $\boldsymbol{h}$:
  all policies beginning with specified actions



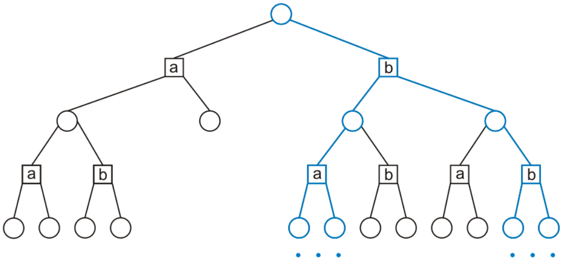
Lower and upper bounds

For any policy $h \in \boldsymbol{h}$, we have $\ell(\boldsymbol{h}) \le v(h) \le b(\boldsymbol{h})$, with:

$$\ell(\boldsymbol{h}) = \sum_{x \in \mathcal{L}(\boldsymbol{h})} \mathrm{P}(x)\, R(x)$$

$$b(\boldsymbol{h}) = \sum_{x \in \mathcal{L}(\boldsymbol{h})} \mathrm{P}(x) \left[R(x) + \frac{\gamma^{d(x)}}{1-\gamma} \right]$$

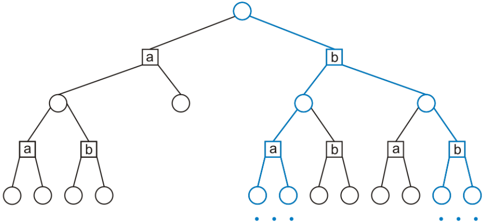
$$= \ell(\boldsymbol{h}) + \sum_{x \in \mathcal{L}(\boldsymbol{h})} \mathrm{P}(x) \frac{\gamma^{d(x)}}{1-\gamma} = \ell(\boldsymbol{h}) + \underbrace{\delta(\boldsymbol{h})}_{\substack{\text{diameter} \\ \text{(uncertainty)}}}$$



Diameter details

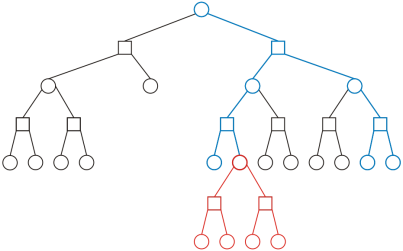
$$\delta(\boldsymbol{h}) = \sum_{x \in \mathcal{L}(\boldsymbol{h})} \underbrace{\mathrm{P}(x) \frac{\gamma^{d(x)}}{1-\gamma}}_{\text{contribution } c(x)}$$

- Generalizes the deterministic-case $\frac{\gamma^d}{1-\gamma}$ = uncertainty due to the single sequence of actions
- Here, uncertainty spread over the policy leaves, each contributing according to its probability and depth



Algorithm: Optimistic planning for MDPs

initialize tree to root node x_0
for $t = 1, \dots, n$ **do**
find optimistic policy $\boldsymbol{h}_t^\dagger = \arg\max_{\boldsymbol{h}} b(\boldsymbol{h})$
expand max-contrib node $x_t^\dagger = \arg\max_{x \in \mathcal{L}(\boldsymbol{h}_t^\dagger)} c(x)$
end for
output near-optimal policy $\boldsymbol{h}^* = \arg\max_{\boldsymbol{h}} \ell(\boldsymbol{h})$



(AISTATS 2012)

Application of **classical AO* search** to MDPs

(Nilsson, 1980)

Near-optimality vs. diameter

For finite sequence $\boldsymbol{h}$, let $v(\boldsymbol{h})$ be the optimal value among sequences starting with $\boldsymbol{h}$.

OP-MDP returns near-optimal policy $\boldsymbol{h}^*$:

$$v^* - v(\boldsymbol{h}^*) \leq \delta^*$$

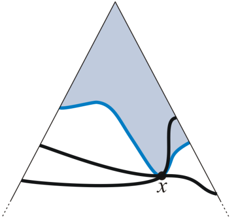
where δ^* is the smallest diameter among all expanded policies

Explored tree

Define **near-optimal tree** $\mathcal{T}_\varepsilon$ containing only the nodes that:

1. have a significant impact: $\alpha(x) \geq \varepsilon$
2. to near-optimal policies: $x \in h$ so that $v^* - v(h) \leq \alpha(x)$

Node impact $\alpha(x)$: *greatest diameter* among policies in which x is *largest-contributing* leaf

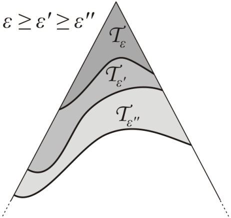


OP-MDP explores $\mathcal{T}_\varepsilon$ so as to always decrease ε; and $\delta^* \leq$ smallest ε seen

Complexity measure

Near-optimality exponent $\beta \geq 0$:

$$|\mathcal{T}_\varepsilon| = \tilde{O}(\varepsilon^{-\beta}) \quad \text{i.e.} \quad |\mathcal{T}_\varepsilon| \leq a(\log 1/\varepsilon)^b \varepsilon^{-\beta} \quad a, b > 0$$



- β describes growth of $\mathcal{T}_\varepsilon$
- Problem is easier when β is smaller:
  - less uniform transition probabilities
  - rewards concentrated on fewer actions

Final guarantee: Near-optimality vs. budget

Theorem

Policy returned is near-optimal:

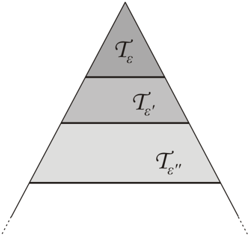
$$v^* - v(\boldsymbol{h}^*) \leq \delta^* = \begin{cases} \tilde{O}(n^{-\frac{1}{\beta}}) & \text{if } \beta > 0 \\ \mathrm{O}(\exp[-(\frac{n}{a})^{\frac{1}{b}}]) & \text{if } \beta = 0 \end{cases}$$

Case 1: Uniform

Identical rewards & uniform probabilities

$$\beta = \frac{\log NM}{\log 1/\gamma} \quad \Rightarrow \quad \delta^* = \tilde{O}(n^{-\frac{\log 1/\gamma}{\log NM}})$$

- $\mathcal{T}_\varepsilon$ grows uniformly, covering full tree
- Algorithm explores this full tree, branching factor NM
- If deterministic $N = 1$, uniform OPD case: $n^{-\frac{\log 1/\gamma}{\log M}}$

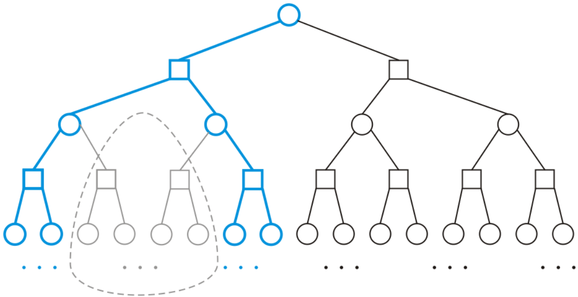


Case 2: Structured rewards

Rewards 1 for one policy h^*, 0 elsewhere; uniform probas

$$\beta = \frac{\log N}{\log 1/\gamma}\left(1 + \frac{\log M}{\log N/\gamma}\right)$$

- $\mathcal{T}_\varepsilon$ grows uniformly in subtree of h^*, with b.f. N (+ some nodes below h^*)
- If $N = 1$, $\beta = 0$, recovering one-path OPD case

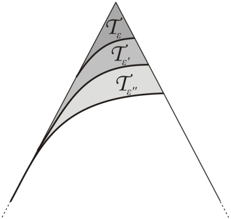


Case 3: Structured probabilities

Identical rewards, Bernoulli probabilities with $p \gg 1 - p$

$$\beta = \frac{\log M\eta}{\log 1/(p\gamma\eta)}$$

- $\mathcal{T}_\varepsilon$ grows in an asymmetric way
- If $p \to 1 \Rightarrow \eta \to 1$, $\beta = \frac{\log M}{\log 1/\gamma}$
  – recover again deterministic case



Using informative bounds

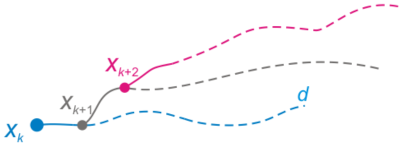
Instead of uninformed bounds 0, $\frac{1}{1-\gamma}$,
use **better bounds** $\underline{V}(x) \leq V^*(x) \leq \overline{V}(x)$ at the leaves:

$$\ell(\boldsymbol{h}) = \sum_{x \in \mathcal{L}(\boldsymbol{h})} \mathrm{P}(x) \left[R(x) + \gamma^{d(x)} \underline{V}(x) \right]$$
$$b(\boldsymbol{h}) = \sum_{x \in \mathcal{L}(\boldsymbol{h})} \mathrm{P}(x) \left[R(x) + \gamma^{d(x)} \overline{V}(x) \right]$$

- Diameters $\delta(\boldsymbol{h})$ decrease, so near-optimality improves ($\ell(\boldsymbol{h}) \leq v(\boldsymbol{h})$ enough, no need for $\ell(\boldsymbol{h}) \leq v(h)\ \forall h \in \boldsymbol{h}$)
- Like for OPD, using ε-accurate upper bound still helps if ε is small enough (EAAI 2016)

Receding horizon control

In practice, work in receding horizon:
apply action u_0 given by $\boldsymbol{h}^*$ at root, then replan



Avoids “running out” of actions,
and compensates for model inaccuracy

HIV treatment

- 6 states:

$$\begin{aligned} T_1, T_2 &\text{ – healthy target cells per ml (types 1 \& 2)} \\ T_1^t, T_2^t &\text{ – infected target cells per ml (types 1 \& 2)} \\ V &\text{ – free virus copies per ml} \\ E &\text{ – immune response cells per ml} \end{aligned}$$

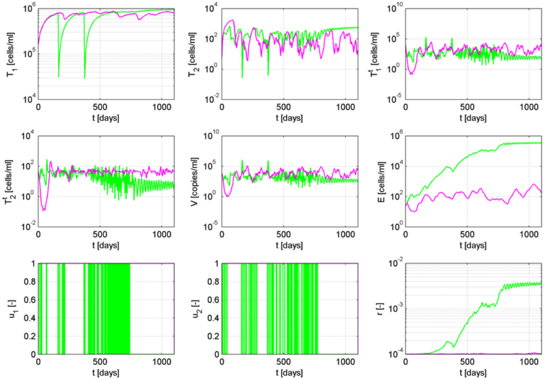
- $M = 2$ actions u_1, u_2: application of RTI and PI drugs
  Random effectiveness among $N = 2$ levels for each drug

Goal: Starting from high level of infection x_0,
optimally switch drugs on and off to:

1. maximize immune response
2. minimize virus load
3. minimize drug use

$$r = c_E E - c_V V - c_1 \epsilon_1 - c_2 \epsilon_2$$

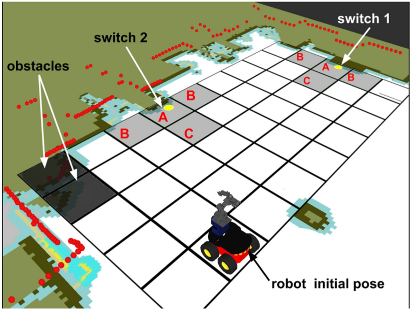
HIV treatment results



- OP vs. full treatment
- Infection eventually controlled **without drugs**

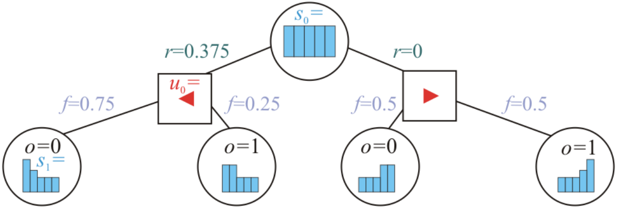
Partially observable MDP

- In a POMDP, the state cannot be measured, instead observations o are made
- After each action u leading to state x', o is observed with probability $O(x', u, o)$
- E.g. robot observes switch states with uncertainty



Solution using planning

- POMDPs often solved via **belief MDP**, with belief state s = proba distribution over underlying states x
- Each action node has N belief children, labeled by observations o and resulting belief x
- Arcs record expected rewards, belief transition probas



Applying OP-MDP

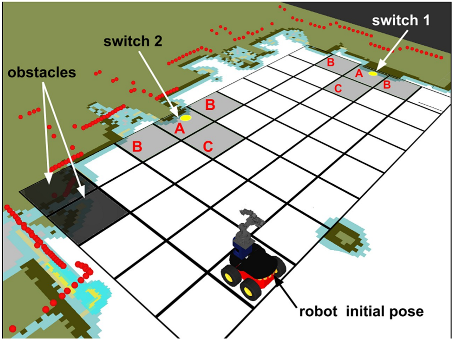
- Apply OP-MDP to explore the tree
  $\Rightarrow$ **AEMS2** algorithm!

(Ross et al., 2007)

- Analysis above directly extends to give convergence rate as a function of POMDP complexity

(IROS 2016)

Example & Demo



- **Objective:** domestic robot makes sure all switches are off
- Fully observable grid position, deterministic NSEW actions
- "Flip" action succeeds stochastically
- Partially observable switch states: "observe" action randomly gives opposite result depending on distance
- Low-level SLAM and control

6 Stochastic case: Optimistic planning for MDPs

7 OP-MDP analysis

8 OP-MDP applications

9 Adversarial case: Optimistic minimax search

10 OMS analysis

11 OMS applications

Adversarial problem

- E.g. if we don't know the next state probas in an MDP, we may **assume the worst possible next states**
- Minimax idea: look for "our" actions u that maximize return assuming opponent takes actions w to minimize it
- Works also for two-player competitive games, robust control, etc.

Problem setting

- Maximizer & minimizer agents,
  with actions $u \in U$ and $w \in W$; $|U| = M, |W| = N$
- They alternately take an infinite sequence of actions:

$$(u_0, w_0, u_1, w_1, \dots) =: (z_0, z_1, z_2, \dots) = \boldsymbol{z}_\infty$$

- Dynamics $x_{d+1} = f(x_d, z_d)$, rewards $r(x_d, z_d)$
- Finite sequence $\boldsymbol{z}_d = (z_0, \dots, z_{d-1})$

Objective

Infinite-horizon value of sequence $\boldsymbol{z}_\infty$:

$$v(\boldsymbol{z}_\infty) := \sum_{d=0}^{\infty} \gamma^d \rho(x_d, z_d).$$

Objective: **discounted minimax-optimal solution:**

$$v^* := \max_{u_0} \min_{w_0} \cdots \max_{u_k} \min_{w_k} \cdots \quad v(\boldsymbol{z}_\infty)$$

Formal setting: Assumptions

Assumptions

- Both agents have discrete actions (as above)
- The rewards $\rho(x, z)$ are in $[0, 1]$ for all $x \in X, z \in U \cup W$.

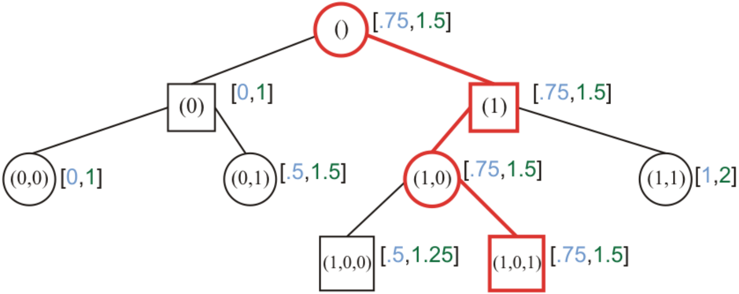
$\Rightarrow$ lower & upper bounds on all sequences $\boldsymbol{z}_\infty$ starting with $\boldsymbol{z}_d$:

$$\ell(\boldsymbol{z}_d) = \sum_{j=0}^{d-1} \gamma^j \rho(x_j, z_j), \quad b(\boldsymbol{z}_d) = \ell(\boldsymbol{z}_d) + \frac{\gamma^d}{1-\gamma} =: \ell(\boldsymbol{z}_d) + \delta(d)$$

where diameter $\delta(d) = \frac{\gamma^d}{1-\gamma}$

Optimistic minimax search

OMS expands tree of possible minmax sequences, using lower and upper bounds on node values



Application of **classical, best-first B* search** to infinite-horizon problems

(Berliner 1979)

Optimistic minimax search (cont'd)

for $t = 1, \ldots, n$ **do**

propagate lower & upper bounds L, B at each node:

$$L(\boldsymbol{z}) \leftarrow \begin{cases} \ell(\boldsymbol{z}), & \text{if } \boldsymbol{z} \text{ leaf} \\ \max / \min_{\boldsymbol{z}' \in \text{children}(\boldsymbol{z})} L(\boldsymbol{z}'), & \text{otherwise} \end{cases}$$

$$B(\boldsymbol{z}) \leftarrow \begin{cases} b(\boldsymbol{z}), & \text{if } \boldsymbol{z} \text{ leaf} \\ \max / \min_{\boldsymbol{z}' \in \text{children}(\boldsymbol{z})} B(\boldsymbol{z}'), & \text{otherwise} \end{cases}$$

choose node to expand: $\boldsymbol{z} \leftarrow$ root, and while not leaf:

$$\boldsymbol{z} \leftarrow \begin{cases} \arg\max_{\boldsymbol{z}' \in \text{children}(\boldsymbol{z})} B(\boldsymbol{z}'), & \text{if } \boldsymbol{z} \text{ max node} \\ \arg\min_{\boldsymbol{z}' \in \text{children}(\boldsymbol{z})} L(\boldsymbol{z}'), & \text{if } \boldsymbol{z} \text{ min node} \end{cases}$$

expand $\boldsymbol{z}$

end for

output a **maximum-depth** expanded node $\boldsymbol{z}^*$

Near-optimality versus diameter

For finite sequence $\boldsymbol{z}$, let $v(\boldsymbol{z})$ be the minimax-optimal value among sequences starting with $\boldsymbol{z}$

If d^* is the largest depth expanded, the solution $\boldsymbol{z}^*$ returned by OMS is $\delta(d^*)$-optimal:

$$|v^* - v(\boldsymbol{z}^*)| \leq \delta(d^*) = \frac{\gamma^{d^*}}{1-\gamma}$$

Note the sequence is already d^* steps long, by definition

Explored tree

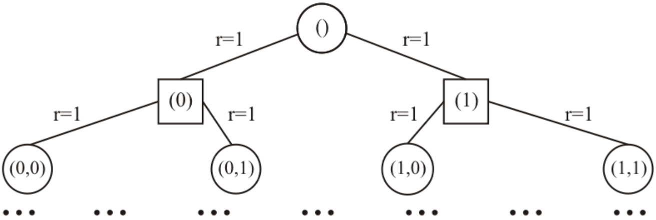
- Algorithm only expands nodes in the subtree:

$$\mathcal{T}^* := \left\{ \boldsymbol{z}_d \,\middle|\, \left| v^* - v(\boldsymbol{z}') \right| \leq \delta(d), \forall \boldsymbol{z}' \text{ on path from root to } \boldsymbol{z}_d \right\}$$

- Intuition: From the information available down to node $\boldsymbol{z}_d$ (interval of values of width $\delta(d) = \frac{\gamma^d}{1-\gamma}$), cannot decide whether the node is (not) optimal. So it must be explored.

Example where the full tree is explored

- All rewards equal to 1, $v^* = \frac{1}{1-\gamma}$
- All solutions have value v^*, so $\mathcal{T}^*$ is the full tree
- $\left|\mathcal{T}_d^*\right| = (MN)^{d/2}$, branching factor $\kappa = \sqrt{MN}$



General case: Branching factor

- Low-complexity special case more involved; in general, branching factor remains a good measure of complexity
- Let $\kappa \in [1, \sqrt{MN}]$ = asymptotic branching factor of $\mathcal{T}^*$
- Problem simpler when κ smaller

Depth vs. budget n

To reach depth d in tree with branching factor κ, we must expand $n = \mathrm{O}(\kappa^d)$ nodes

$$\Rightarrow \quad d^* = \Omega(\frac{\log n}{\log \kappa})$$

Final guarantee: Near-optimality vs. budget

Theorem

Given budget n, we have:

$$|v^* - v(\boldsymbol{z}^*)| \leq \delta(d^*) = \frac{\gamma^{d^*}}{1-\gamma} = \begin{cases} \mathrm{O}(n^{-\frac{\log 1/\gamma}{\log \kappa}}) & \text{if } \kappa > 1 \\ \mathrm{O}(\gamma^{n/C}) & \text{if } \kappa = 1 \end{cases}$$

- Faster convergence when κ smaller (simpler problem)
- Exponential convergence when $\kappa = 1$

Using informative bounds

Instead of uninformed bounds $0, \frac{1}{1-\gamma}$, use **better bounds** $\underline{V}(x) \leq V(x) \leq \overline{V}(x)$ on minimax value $V(x)$ at leaf states:

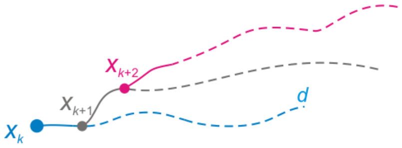
$$\ell(\boldsymbol{z}_d) = \sum_{j=0}^{d-1} \gamma^j \rho(x_j, z_j) + \gamma^d \underline{V}(x_d)$$

$$b(\boldsymbol{z}_d) = \sum_{j=0}^{d-1} \gamma^j \rho(x_j, z_j) + \gamma^d \overline{V}(x_d)$$

- Diameters δ decrease, so near-optimality improves

Receding horizon control

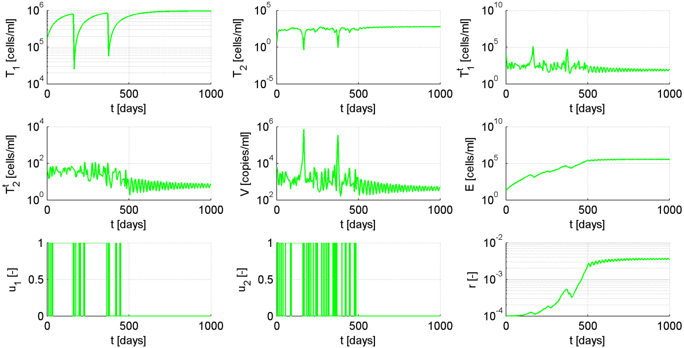
In practice, work in receding horizon:
apply first max action u_0 on sequence $\boldsymbol{z}^*$ returned, then replan



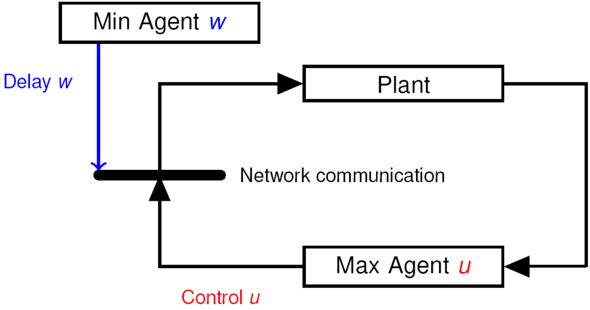
HIV: OMS results

Random disturbance treated as opponent

Budget of $n = 4000$ node expansions



Switched control over delayed network



- Max action = controlled “mode”
  e.g. constant action or low-level controller
- Min action = network delay

Quanser inverted pendulum

System:

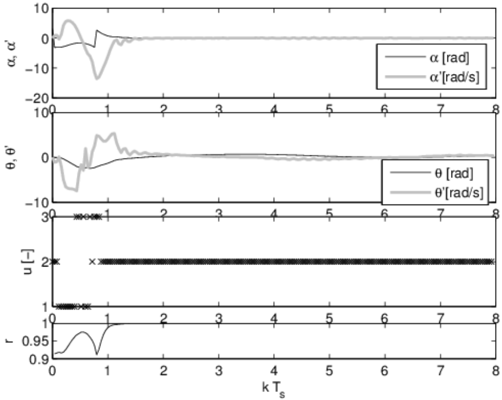
- x = rod angle α, base angle θ, angular velocities
- input ω = voltage
- Sampling time $T_s = 0.04$

Goal: swing up & stabilize pointing up:

- $\rho = -15\alpha^2 - 0.05(\theta^2 + \dot{\alpha}^2 + \dot{\theta}^2 + \omega^2)$, normalized to $[0, 1]$
- Discount factor $\gamma = \sqrt{0.95}$

Results

- 3 modes: #1 constant -6 V, #3 constant 6 V, #2 a stabilizing mode $\omega = Kx$ computed with LQR
- 2 delays: 0 or 1 steps
- Use real-time framework like OPD, plan during entire T_s



References for Part II

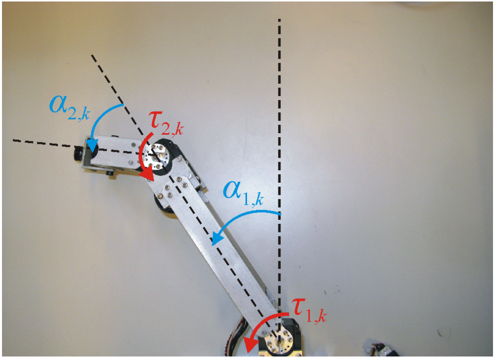
- Berliner, *The B* Search Algorithm: A Best First Proof Procedure*, Artificial Intelligence 1979.
- Nilsson, *Principles of Artificial Intelligence*, 1980.
- Ross et al., *AEMS: An anytime online search algorithm for approximate policy refinement in large POMDPs*, IJCAI 2007.
- Busoniu, Munos, *Optimistic Planning for Markov Decision Processes*, AISTATS 2012.
- Busoniu, Daniels, Babuska, *Online Learning for Optimistic Planning*, Engineering Applications of AI, 2016.
- Pall, Tamas, Busoniu, *An Analysis and Home Assistance Application of Online AEMS2 Planning*, IROS 2016.

Part III

Continuous-action MDPs

Continuous actions

In control applications, *u* often **continuous**! E.g. robot arm:



Scalar actions in this talk, although algorithms can be extended to vector actions (at significantly larger computational cost)

Assumptions

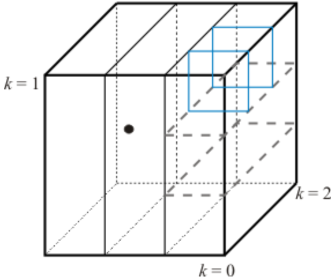
- Rewards $r \in [0, 1]$
- Scalar compact action space $U = [0, 1]$
- Lipschitz-continuous dynamics and rewards:

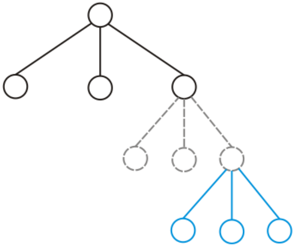
$$\left\|f(x,u) - f(x',u')\right\| \leq L_f(\left\|x - x'\right\| + \left|u - u'\right|)$$
$$\left|\rho(x,u) - \rho(x',u')\right| \leq L_\rho(\left\|x - x'\right\| + \left|u - u'\right|)$$

- $\gamma L_f < 1$: most restrictive

Search refinement

- Split U^∞ iteratively, leading to a tree of hyperboxes





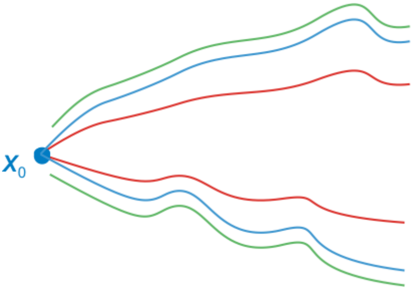
- Each box i only represents explicitly dimensions already split, $k = 0, \dots, K_i - 1$
- Box i has value $v(i) = \sum_{k=0}^{K_i-1} \gamma^k r_{i,k+1}$, rewards of center sequence

Lipschitz value function

- For any two action sequences $\boldsymbol{u}_\infty, \boldsymbol{u}'_\infty$:

$$\left| v(\boldsymbol{u}_\infty) - v(\boldsymbol{u}'_\infty) \right| \leq \frac{L_\rho}{1 - \gamma L_f} \sum_{k=0}^{\infty} \gamma^k \left| u_k - u'_k \right|$$

- Intuition: states (and so rewards) may diverge somewhat, but divergence controlled due to $\gamma L_f < 1$

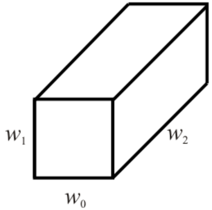


Box upper bound

- For any sequence $\boldsymbol{u}_\infty$ in box i:

$$v(\boldsymbol{u}_\infty) \leq v(i) + \frac{\max\{1, L_\rho\}}{1 - \gamma L_f} \sum_{k=0}^{\infty} \gamma^k w_{i,k} := b(i)$$

- $w_{i,k}$ width of dimension k, 1 if not split yet



- $b(i)$ **b-value** of box i

Diameter and dimension selection

- **Diameter** $\delta(i) := \frac{\max\{1, L_\rho\}}{1-\gamma L_f} \sum_{k=0}^{\infty} \gamma^k w_{i,k}$
  $=$ uncertainty on values in the box
- **Impact** of dimension k on uncertainty is $\gamma^k w_{i,k}$

$\Rightarrow$ when splitting a box, choose dimension with largest impact, to reduce uncertainty the most

- Always split into odd $M > 1/\gamma$ pieces

OPC algorithm

Optimistic planning with continuous actions (OPC)

initialize tree with root box U^∞
while budget of model calls n not exhausted **do**
select **optimistic** leaf box $i^\dagger = \arg\max_{i \in \mathcal{L}} b(i)$
select **max-impact** dimension $k^\dagger = \arg\max_k \gamma^k w_{i^\dagger,k}$
split $i^\dagger$ along $k^\dagger$, creating M children on the tree
end while
return best center sequence seen, $i^* = \arg\max_i v(i)$

(ACC 2016)

Computation measured by model calls (f, ρ) instead of node expansions, since an expansion simulates sequences of varying lengths, at varying computational costs

Near-optimality vs. diameter

OPC returns a sequence i^* that is near-optimal:

$$v^* - v(i^*) \leq \delta^*$$

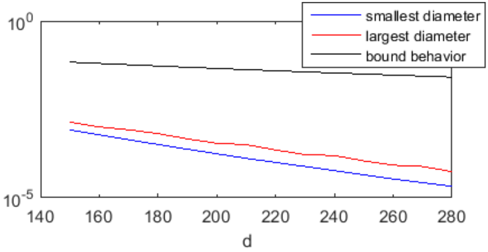
where δ^* is the smallest diameter of any expanded node

Diameter vs. depth

Given depth in tree d =total number of splits:

$$\delta(i) = \tilde{O}(\gamma^{\sqrt{2d^{\frac{\tau-1}{\tau^2}}}}), \text{ where } \tau = \left\lceil \frac{\log 1/M}{\log \gamma} \right\rceil$$

Diameters vary by the order of splits, but they all converge to 0 roughly exponentially in $\sqrt{d}$. Example:



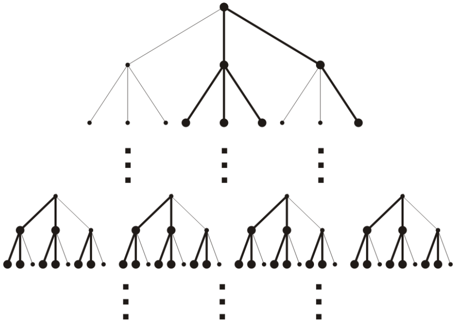
Branching factor

- OPC only expands in near-optimal subtree:

$$\mathcal{T}^* = \{i \in \mathcal{T} \mid v^* - v(i) \leq \delta(i)\}$$

- Special cases rather complicated, but asymptotic branching factor $\kappa \in [1, M]$ of $\mathcal{T}^*$ remains good **problem complexity measure**

E.g. $\kappa = 2$, $M = 3$:



Depth vs. budget n

To reach depth d in tree with branching factor κ,
we must expand $\mathrm{O}(\kappa^d)$ **nodes**,
which takes $n = \mathrm{O}(d\kappa^d) = \tilde{\mathrm{O}}(\kappa^d)$ **model calls**

$$\Rightarrow \text{ largest depth } d^* = \tilde{\Omega}\left(\frac{\log n}{\log \kappa}\right)$$

Final guarantee: Near-optimality vs. budget

Theorem

After spending n model calls, OPC suboptimality is:

$$v^* - v(i^*) \leq \delta^* \leq \delta(d^*) = \begin{cases} \tilde{O}(\gamma^{\sqrt{\frac{2(\tau-1)\log n}{\tau^2 \log \kappa}}}), & \text{if } \kappa > 1 \\ \tilde{O}(\gamma^{n^{1/4} b}), & \text{if } \kappa = 1 \end{cases}$$

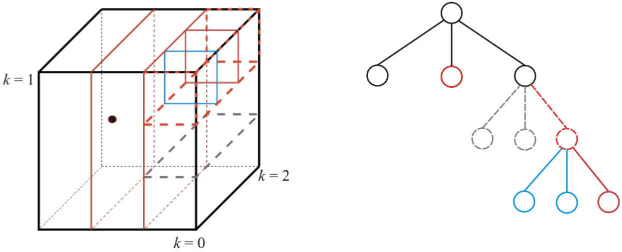
- Convergence faster when κ smaller
- When $\kappa = 1$, convergence is exponential in power $n^{1/4}$
- When $\kappa > 1$, we pay for generality: exponential computation κ^d to reach depth d

Idea

- Avoid using Lipschitz constants (i.e. diameters) altogether

$\Rightarrow$ Split a **potentially optimistic** box at each depth:

$$i_d^\dagger = \arg\max_{i \text{ at } d} v(i); \text{ proxy for unknown } b(i) = v(i) + \delta(i)$$



- Depth cutoff at $d_{\max}(n)$ to avoid indefinite expansion

SOPC algorithm

initialize tree with root box
while n not exhausted **do**
  for d = first unexpanded to $d_{\max}(n)$ **do**
    potentially optimistic leaf $i_d^\dagger = \arg\max_{i \in \mathcal{L}_d} v(i)$
    max-impact dimension $k_d^\dagger = \arg\max_k \gamma^k w_{i_d^\dagger, k}$
    split $i_d^\dagger$ along $k_d^\dagger$
  end for
end while
return best sequence seen $i^* = \arg\max_i v(i)$

Depth vs. budget n

SOPC may expand outside $\mathcal{T}^*$ but not too much
After spending n it reaches d^* where:

$$n = \mathrm{O}(d^2_{\max}(n) \sum_{k=1}^{d^*} \kappa^k)$$

(or $d_{\max}(n)$ if it is smaller)

Performance guarantee

Theorem

For budget n, SOPC suboptimality is:

$$v^* - v(i^*) = \begin{cases} \tilde{O}\left(\gamma^{\sqrt{\frac{2(1-2\varepsilon)(\tau-1)\log n}{\tau^2 \log \kappa}}}\right), & \text{if } \kappa > 1 \text{ and } d_{\max}(n) = n^{\varepsilon} \\ \tilde{O}\left(\gamma^{n^{1/6} b}\right), & \text{if } \kappa = 1 \text{ and } d_{\max}(n) = n^{1/3} \end{cases}$$

- When $\kappa > 1$, with small ε nearly same bound as OPC
- When $\kappa = 1$, $n^{1/6}$ instead of $n^{1/4}$ – slower but similar
- All this while **adapting to unknown smoothness**

Recall: Quanser pendulum

System:

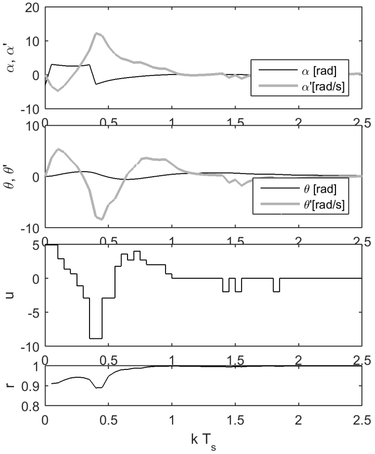
- x = rod angle α, base angle θ, angular velocities
- Input ω = voltage
- Sampling time $T_s = 0.05$

Goal: swing up & stabilize pointing up:

- $\rho = -\alpha^2 - \theta^2 - .005(\dot{\alpha}^2 + \dot{\theta}^2) - .05u^2$, normalized to $[0, 1]$
- Discount factor $\gamma = 0.85$

Controlled trajectory

$n = 5000$ model calls; note adaptive discretization of control magnitude, and no access to stabilizing mode



Real-time control

Uses parallelized real-time framework similar to OPD

Other optimistic planners

- Search open-loop sequences in stochastic MDPs:
  OLOP (Bubeck & Munos, 2010)
- Learn the MDP model while searching:
  BOP (Fonteneau et al., 2013)
- Sample-based continuous-action planning (Mansley et al., 2010)
- etc.

Related fields

Monte Carlo tree search

- Selects leaf to expand according to bandit UCBs; prototypical algorithm UCT
- Estimates leaf values by running long random simulations

(Browne et al., 2012)

Planning and scheduling

- Different formalism but algorithms often applicable to MDPs

Nonlinear model-predictive control

- Focus on stability and exploiting dynamics knowledge

(Grune & Pannek, 2016)

Nonlinear control applications

- Switched systems = natural discrete-action MDPs

  (Automatica 2017)

- Nonlinear networked control via sequences

  (TAC 2016)

- Cooperative control in multiagent systems

  (CTT 2015)

Conclusion

Optimistic planning

Online model-based, good convergence guarantees

Works for complex dynamics & states, but simple actions

Thank you!

References for Part III

- Bubeck, Munos, *Open Loop Optimistic Planning*, COLT 2010.
- Mansley et al., *Sampled-Based Planning for Continuous-Action MDPs*, ICAPS 2011.
- Fonteneau et al., *Optimistic Planning for Belief-Augmented MDPs*, ADPRL 2013.
- Browne et al., *A Survey of Monte-Carlo Tree Search Methods*, IEEE Trans on Computational Intelligence and AI in Games 2012.
- Grune, Pannek, *Nonlinear Model-Predictive Control*, 2016.
- Busoniu, Morarescu, *Topology Preserving Flocking of Nonlinear Agents using Optimistic Planning*, Control Theory & Technology 2015.
- Busoniu, Pall, Munos, *Discounted Near-Optimal Control of General Continuous-Action Nonlinear Systems [...]*, ACC 2016.
- Busoniu, Postoyan, Daafouz, *Near-Optimal Strategies for Nonlinear and Uncertain Networked Control Systems*, IEEE TAC 2016.
- Busoniu, Daafouz, Bragagnolo, Morarescu, *Planning [...] in Nonlinear Systems with Controlled or Uncontrolled Switches*, Automatica 2017.