Approximate dynamic programming and reinforcement learning for control

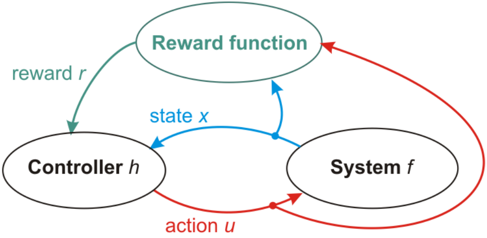
Lucian Buşoniu

Universitat Politècnica de València, 21-23 June 2017

Part III

Optimistic planning

Recall: Deterministic problem



- Observe states x, apply actions u, receive rewards r
- System: dynamics $x_{k+1} = f(x_k, u_k)$
- Performance: reward function $r_{k+1} = \rho(x_k, u_k)$
- **Objective**: maximize discounted return $\sum_{k=0}^{\infty} \gamma^k r_{k+1}$, discount factor $\gamma \in (0, 1)$

Part III in course structure

- Problem definition. Discrete-variable exact methods
- Continuous-variable, approximation-based methods
- **Optimistic planning**

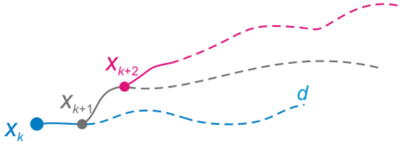
Methods presented so far are the main ones in the field
In this part, **current research** topic in the ROCON group at Cluj.

Online planning idea

At each step k, solve local optimal control at state x_k:

- Infinite action sequences: $\boldsymbol{u}_\infty = (u_k, u_{k+1}, \dots)$
- Optimization problem: $\sup_{\boldsymbol{u}_\infty} v(\boldsymbol{u}_\infty)$ $(= \sum_{i=0}^{\infty} \gamma^i r_{k+1+i})$

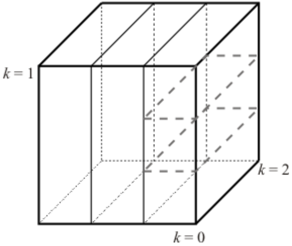
1. Explore sequences from x_k, to find a near-optimal one
2. Apply first action of this sequence, and repeat



Receding-horizon model-predictive control

Optimistic planning (OP) idea

initialize **set of all possible sequences**
repeat
  select most promising, **optimistic** set
  refine selected set
until computation budget n exhausted
return sequence in best set



Advantages of OP

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

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

- ...for general nonlinear dynamics and rewards
- Since it reruns at each state, no direct dependence on state space size – continuous states not a problem

Algorithm landscape

By model usage:

- Model-based: f, ρ known
- Model-free: f, ρ unknown (reinforcement learning)

By interaction level:

- Offline: algorithm runs in advance
- Online: algorithm runs with the system

Exact vs. approximate:

- Exact: x, u small number of discrete values
- Approximate: x, u continuous (or many discrete values)

Problem setting

Assumptions

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

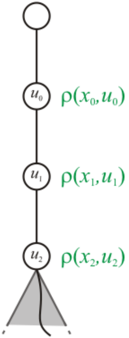
- Again, continuous states handled natively
- If actions continuous $\Rightarrow$ must be discretized

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 $\boldsymbol{u}_\infty \in \boldsymbol{u}_d$
- $b(\boldsymbol{u}_d) = \ell(\boldsymbol{u}_d) + \frac{\gamma^d}{1-\gamma}$
  upper bound on returns of $\boldsymbol{u}_\infty \in \boldsymbol{u}_d$

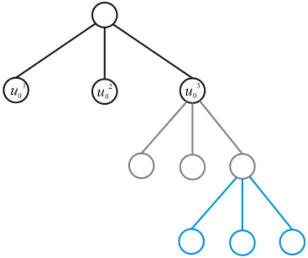
- $v(\boldsymbol{u}_d) = \sup_{\boldsymbol{u}_\infty \in \boldsymbol{u}_d} v(\boldsymbol{u}_\infty)$
  value of applying $\boldsymbol{u}_d$ and then acting optimally



Algorithm: OPD

Optimistic planning for deterministic systems (OPD)

initialize empty sequence $\boldsymbol{u}_0$ (= all infinite sequences)
loop n times
  select **optimistic** leaf sequence $\boldsymbol{u}_d^\dagger$, maximizing b
  expand $\boldsymbol{u}_d^\dagger$: initialize all values for the $d+1$-th action
end loop
return greedy $\boldsymbol{u}_{d^*}^*$ maximizing ℓ



Near-optimality vs. depth

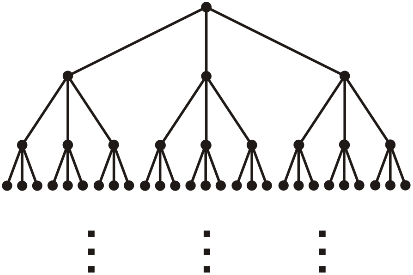
1. OPD returns a sequence $\boldsymbol{u}^*_{d^*}$, with length $d^* =$ the deepest expanded d
2. This sequence is near-optimal:

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

where v^* the optimal value (at x_0)

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

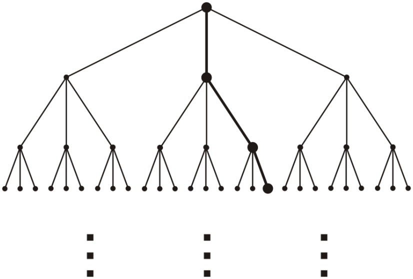
So to expand all nodes down to depth d, we must spend:

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

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

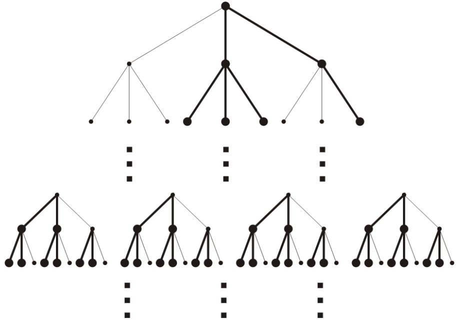
General case: Branching factor

- Algorithm only expands in near-optimal subtree:

$$\mathcal{T}^* = \left\{ \boldsymbol{u}_d \;\middle|\; v^* - v(\boldsymbol{u}_d) \leq \frac{\gamma^d}{1-\gamma} \right\}$$

- Define κ = asymptotic branching factor of $\mathcal{T}^*$:
  problem complexity measure, $\kappa \in [1, K]$

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

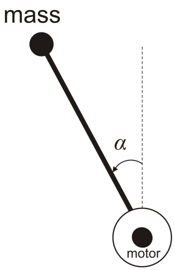
Theorem

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

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

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

Recall: Inverted pendulum swing-up



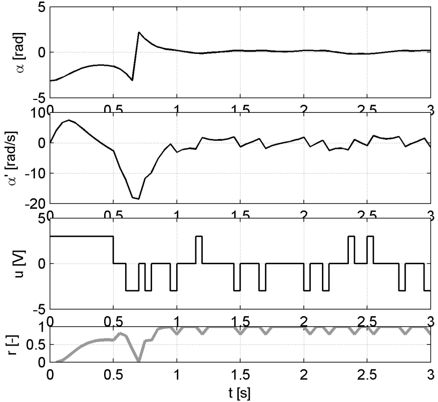
- $x = [\alpha, \dot{\alpha}]^\top$, u = voltage
- Stabilize pointing up, requires swing-up

Challenging for planning:
long trajectories, misleading short-term rewards

Simulation: Inverted pendulum demo

Demo

Swingup trajectory:

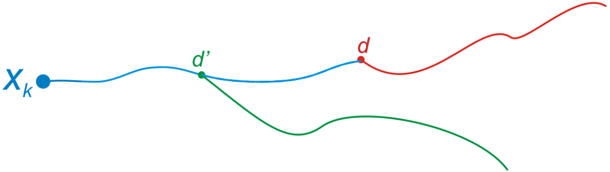


Real-time idea

Challenge: computation time large and must be handled!

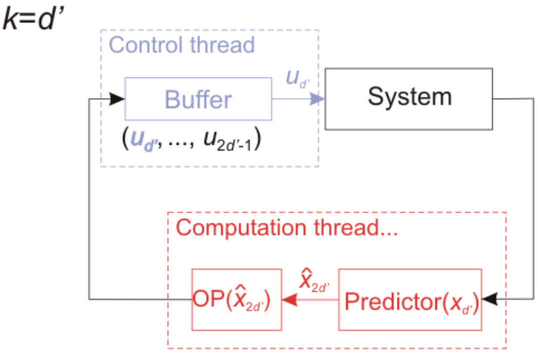
- Usually only first action of each sequence is sent to actuator
- But remember: OP returns **long sequences**!

$\Rightarrow$ 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



Setting up real-time OPD

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

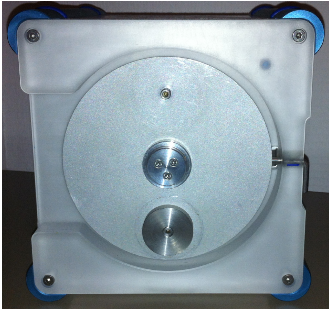
$\Rightarrow$ Select subsequence length d' so that:

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

- 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



Assumptions

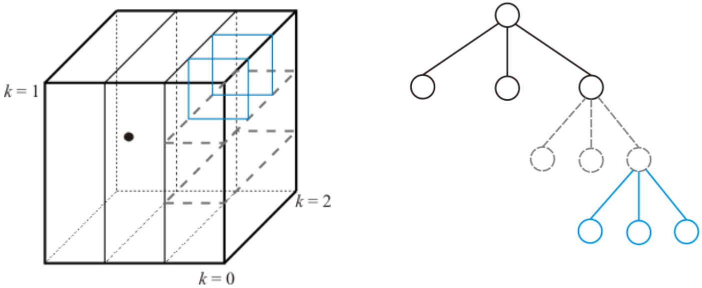
- Rewards $r \in [0, 1]$
- Scalar continuous action space $U = [0, 1]$
  (can be extended to vector actions)
- Lipschitz-continuous dynamics and rewards:

$$\begin{aligned} \left\| f(x, u) - f(x', u') \right\| &\leq L_f(\left\| x - x' \right\| + \left| u - u' \right|) \\ \left| \rho(x, u) - \rho(x', u') \right| &\leq L_\rho(\left\| x - x' \right\| + \left| u - u' \right|) \end{aligned}$$

- $\gamma L_f < 1$: most restrictive

Search refinement

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



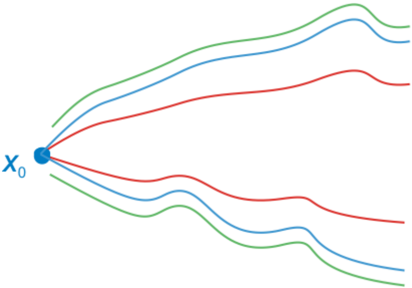
- Each box i only represents explicitly dimensions already split, $k = 0, \dots, K_i - 1$
- Box i has value $v(i) = \sum_{k=0}^{K_i - 1} \gamma^k r_{i,k+1}$, rewards of center sequence

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 $\gamma L_f < 1$

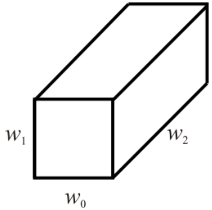


Box upper bound

- 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 $T > 1/\gamma$ pieces

OPC algorithm

Optimistic planning with continuous actions (OPC)

Input: budget of **model calls** n

initialize tree with root box U^∞

while 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 T children on the tree

end while

return best center sequence seen, $i^* = \arg\max_i v(i)$

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:

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

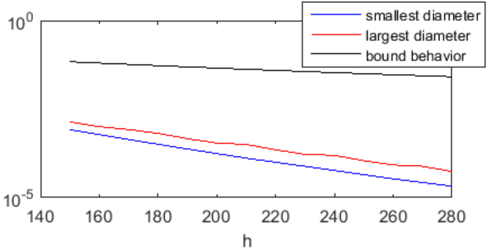
where δ^* is the smallest diameter of any expanded node

Diameter vs. depth

Given depth in tree d =total number of splits:

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

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



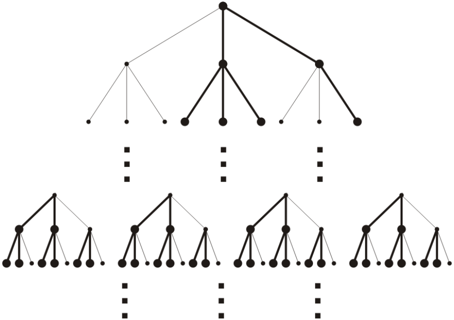
Branching factor

- OPC only expands in near-optimal subtree:

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

- Special cases more complicated than OPD, but asymptotic branching factor $t \in [1, T]$ of $\mathcal{T}^*$ remains good **problem complexity measure**

E.g. $t = 2$, $T = 3$:



Depth vs. budget n

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

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

Final guarantee: Near-optimality vs. budget

Theorem

After spending n model calls, OPC suboptimality is:

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

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

Inverted pendulum demo

Note different variant of the algorithm called ‘simultaneous’ OPC, with nearly the same guarantees

Demo

Quanser pendulum

System:

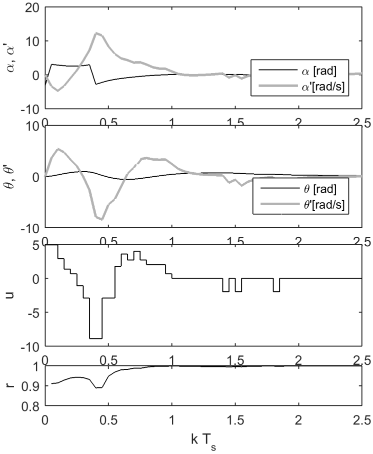
- x = rod angle α, base angle θ, angular velocities
- u = motor voltage $\in [-9, 9]$ V
- Sampling time $T_s = 0.05$

Goal: stabilize pointing up:

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

Controlled trajectory

$n = 5000$ model calls; note adaptive discretization of control magnitude



Real-time control

Uses the same parallelized real-time framework as OPD

Real-time demo

Problem setting

- Maximizer & minimizer agents,
  with actions $u \in U$ and $w \in W$; $|U| = N_U, |W| = N_W$
- 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 $r(x_d, z_d)$
- Denote 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:

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

Main assumption

Assumption

The rewards $\rho(x, z)$ are in $[0, 1]$ for all $x \in X, z \in U \cup W$.

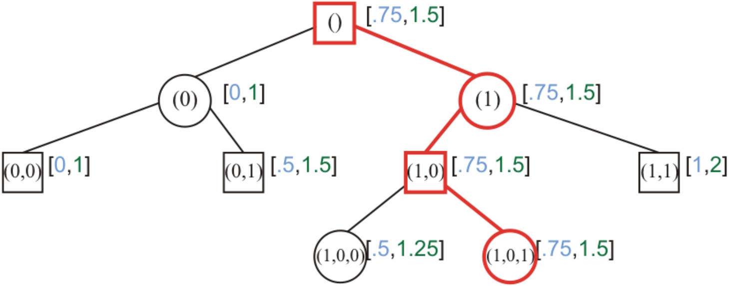
$\Rightarrow$ lower & upper bounds on all sequences $\boldsymbol{z}_\infty$ starting with $\boldsymbol{z}_d$:

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

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

OMS algorithm

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



Natural application of optimistic principle, and already known since $\sim$1980 as best-first B* search

OMS algorithm (cont'd)

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

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

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

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

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 $\widehat{\boldsymbol{z}}$

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 $\widehat{\boldsymbol{z}}$ returned by OMS is $\delta(d^*)$-optimal:

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

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

Explored tree

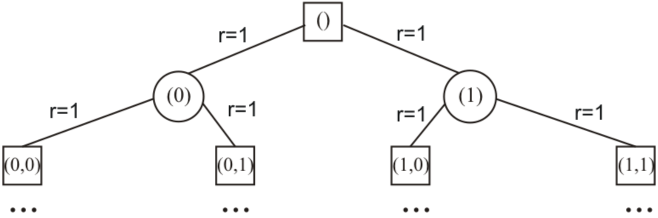
- Algorithm only expands nodes in the subtree:

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

- 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
- $\left|\mathcal{T}_d^*\right| = (N_U N_W)^{d/2}$, branching factor $\beta = \sqrt{N_U N_W}$



General case: Branching factor

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

Depth vs. budget n

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

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

Final guarantee: Near-optimality vs. budget

Theorem

Given budget n, we have:

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

- Faster convergence when β smaller (simpler problem)
- Exponential convergence when $\beta = 1$

HIV infection treatment

- 6 states:

  T_1, T_2, T_1^t, T_2^t – healthy & infected target cells / ml (type 1 & 2)

  V, E – free virus copies & immune response cells / ml
- 2 binary actions u_1, u_2: application of RTI and PI drugs
- **Disturbance:** stochastic drug effectiveness

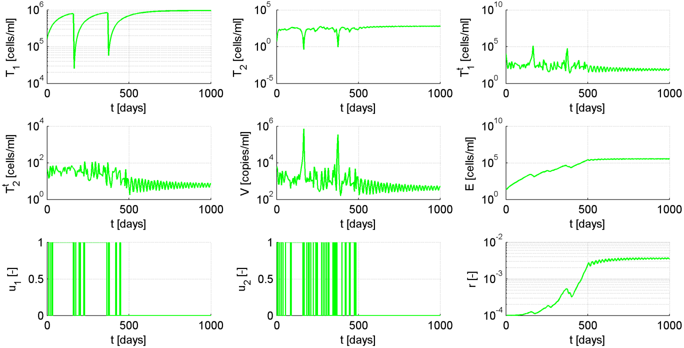
Goal: Starting from high level of infection x_0, optimally switch drugs on and off to:

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: OMS results

Budget of $n = 4000$ node expansions



Infection eventually controlled without drugs

Open issues

RL & DP **active research fields**

Open problems:

- Approximator design
- Data efficiency
- High-dimensional states and actions
- Unmeasurable states
- Safety and stability guarantees

Summary

RL, DP, and planning =

Near-optimal control of **general nonlinear, possibly unknown** systems

References for Part III

- 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.
- Busoniu, Pall, Munos, *Discounted Near-Optimal Control of General Continuous-Action Nonlinear Systems Using Optimistic Planning*, ACC 2016.
- Busoniu, Pall, Munos, *An analysis of optimistic, best-first search for minimax sequential decision making*, ADPRL 2014.

+ control applications: TAC'16, Automatica'17, ACC'17, etc.