# Planning Methods for Near-Optimal Nonlinear Control

Lucian Buşoniu

Automation Department
Technical University of Cluj-Napoca, Romania
Contact: lucian@busoniu.net

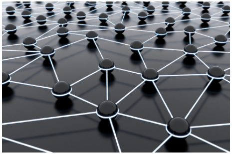
30 June 2016, City University London

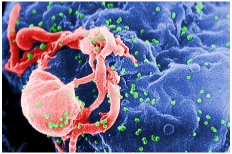
# Overall theme

**AI-based control of complex systems**

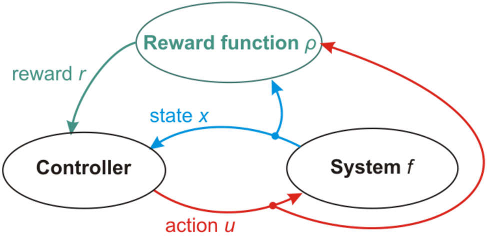
Complexity: nonlinearity, stochastic dynamics, unknown behavior, distributed structure, . . .

Applications: robotics, control, medicine, . . .



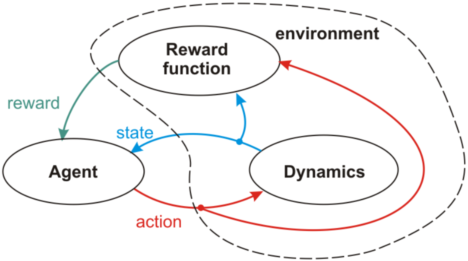


# Optimal control problem (deterministic MDP)




- 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**: maximize discounted return $\sum_{k=0}^{\infty} \gamma^k r_{k+1}$, 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: Quanser pendulum

**System**:

- $x$ = rod angle $\alpha$, base angle $\theta$, 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

# Solution methods

By path to solution:

- Policy iteration: find the value function of current control policy, use it to improve the policy, repeat to convergence
- Value iteration: find optimal value function, use it to compute optimal policy
- Policy search: look directly for the optimal policy

By model usage:

- Model-based: dynamics and reward function known
- Model-free: only data (reinforcement learning)

By interaction level:

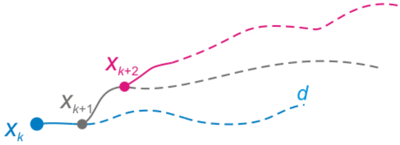
- Offline: algorithm is run in advance
- Online: algorithm runs while controlling system

# Online planning

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: $\sup_{\boldsymbol{u}_\infty} v(\boldsymbol{u}_\infty)$ $(= \sum_{i=0}^{\infty} \gamma^i r_{k+1+i})$

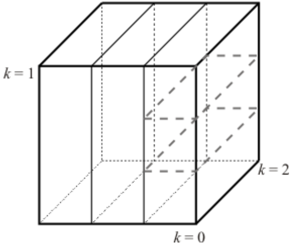
1. Explore sequences from $x_k$, to find a near-optimal one
2. Apply first action of this sequence, and repeat




Receding-horizon model-predictive control

# Optimistic planning (OP): Main 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




Bandit-based optimization; branch & bound if deterministic

# Advantages of OP

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

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

(Munos, 2014)

- ...for general nonlinear dynamics and rewards

1. Introduction

2. OPC: Optimistic planning with continuous actions
   - Setting and algorithm
   - Analysis
   - Experiments

3. SOPC: Simultaneous OPC
   - Algorithm and guarantee
   - Experiments

4. Conclusions

# Assumptions

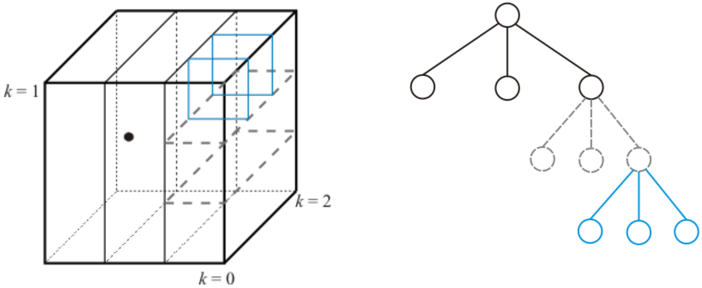
- Rewards $r \in [0, 1]$
- Action space $U = [0, 1]$
  (can be extended to compact multidimensional $U$)
- Lipschitz dynamics and rewards:

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

- $\gamma L_f < 1$: most restrictive

# Search refinement

- Split $U^\infty$ iteratively, leading to a tree of hyperboxes



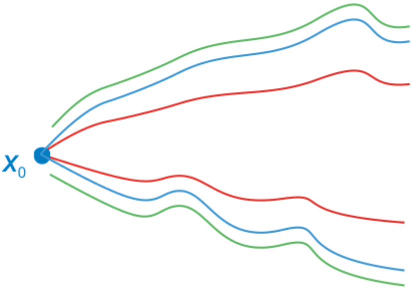

- Each box $i$ only represents explicitly dimensions already split, $k = 0, \dots, K_i - 1$
- Box $i$ has value $v(i) = \sum_{k=0}^{K_i-1} \gamma^k r_{i,k+1}$, rewards of center sequence

# 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 $\gamma L_f < 1$

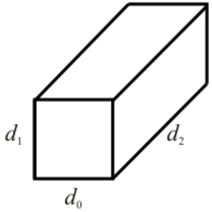


# Box upper bound

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

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

- $d_{i,k}$ length of dimension $k$, 1 if not split yet




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

# Diameter and dimension selection

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

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

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

# OPC algorithm

initialize tree with root box $U^\infty$
**while** $n$ not exhausted **do**
  select **optimistic** leaf box $i^\dagger = \arg\max_{i \in \mathcal{L}} b(i)$
  select **max-impact** dimension $k^\dagger = \arg\max_k \gamma^k d_{i^\dagger,k}$
  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)

1. Introduction

2. OPC: Optimistic planning with continuous actions
   - Setting and algorithm
   - Analysis
   - Experiments

3. SOPC: Simultaneous OPC

4. Conclusions

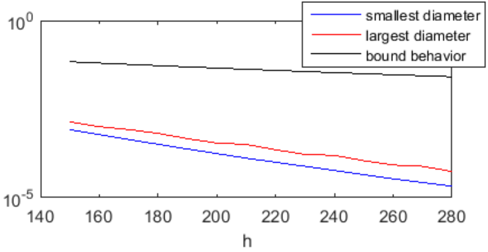
# Diameter bound

## Lemma

Given depth in tree $h =$total number of splits:

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

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



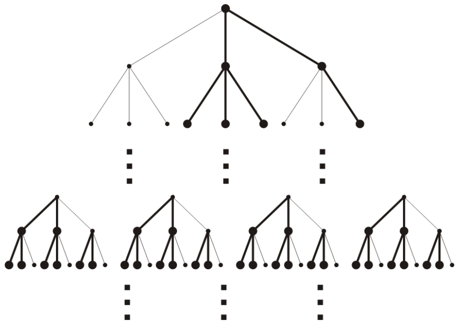
# Complexity measure

- OPC only expands in near-optimal subtree:

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

  (nodes that cannot be eliminated as suboptimal)
- Define $m \in [1, M]$ = asymptotic branching factor of $\mathcal{T}^*$: **problem complexity measure**

E.g. $m = 2$, $M = 3$:



# Performance guarantee

## Theorem

After spending $n$ model calls, OPC suboptimality is:

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

- Convergence rate modulated by problem complexity $m$, faster when $m$ smaller
- When $m = 1$, convergence is fast, with power $n^{1/4}$
- When $m > 1$, we pay for generality: exponential computation $m^h$ to reach depth $h$

# Related work

- Inspired by optimistic optimization (DOO & SOO)

  (Munos 2011)

- HOLOP, HOOT: fixed finite horizon

  (Weinstein et al. 2012, Mansley et al. 2011)

- Lipschitz planning (LP): different dimension selection, no guarantees

  (Hren 2012)

- SOOP: no Lipschitz constants, no guarantees

  (ADPRL 2013)

1. Introduction

2. OPC: Optimistic planning with continuous actions
   - Setting and algorithm
   - Analysis
   - Experiments

3. SOPC: Simultaneous OPC

4. Conclusions

# Recall example: Quanser pendulum

System:

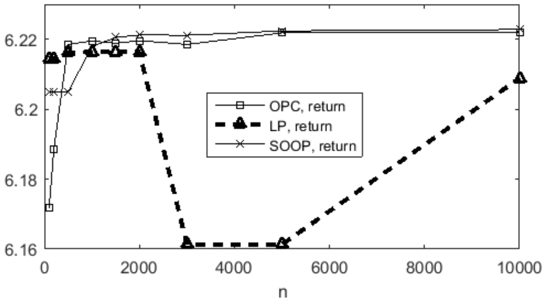
- $x$ = rod angle $\alpha$, base angle $\theta$, angular velocities
- $u$ = motor voltage $\in [-9, 9]$ V
- Sampling time $T_s = 0.05$

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

# OPC versus LP and SOOP

OPC with $L_f = L_\rho = 1.1$, tighter diameter formula
Parameters of algorithms optimized



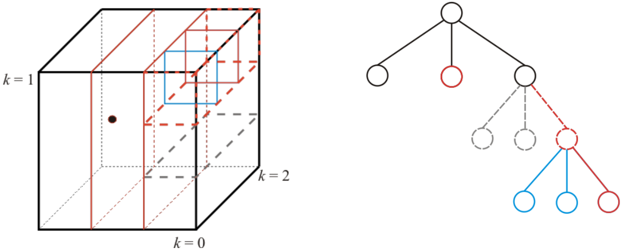

$\Rightarrow$ OPC theoretically nice, beaten by heuristic SOOP in practice
Disadvantage: **under/overestimated Lipschitz constants**

# Idea

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

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

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




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

# SOPC algorithm

initialize tree with root box
**loop**
  **for** $h$ = first unexpanded to $h_{\max}(n)$ **do**
    **potentially optimistic** leaf $i_h^\dagger = \arg\max_{i \in \mathcal{L}_h} v(i)$
    max-impact dimension $k_h^\dagger = \arg\max_k \gamma^k d_{i_h^\dagger, k}$
    split $i_h^\dagger$ along $k_h^\dagger$
    **if** budget $n$ reached, **stop algo end if**
  **end for**
**end loop**
**return** best sequence seen $i^* = \arg\max_i v(i)$

# Performance guarantee

**Theorem**

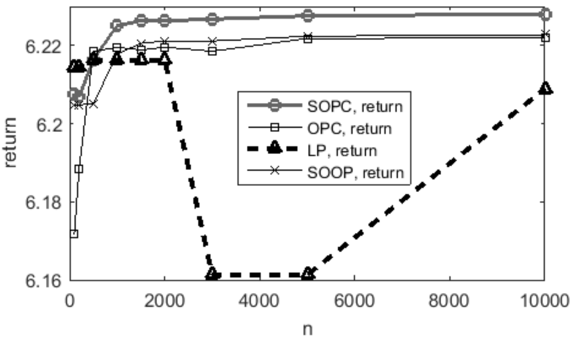
For budget $n$, SOPC suboptimality is:

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

- When $m > 1$, with small $\varepsilon$ nearly same bound as OPC
- Intuition: expanding full path takes up to $h_{\max}(n)$, negligible compared to exponential tree size $m^h$
- When $m = 1$, $n^{1/6}$ instead of $n^{1/4}$ – slower but similar
- All this while **adapting to unknown smoothness**

# Quanser pendulum results

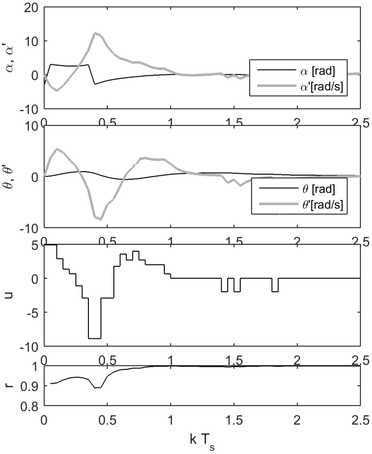
SOPC with $h_{\max}(n) = n^{0.45}$




$\Rightarrow$ best algorithm

# Controlled trajectory

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



# Real-time control

# Conclusions

(Simultaneous) optimistic planning with continuous actions:

- Control of general nonlinear systems,
  guaranteed near-optimal
- SOPC adapts to unknown smoothness,
  works well in practice

Next steps:

- Eliminate assumption $\gamma L_f < 1$ using stability
- Reduce complexity while still keeping problem interesting?

Thank you!