17.2.2015

Optimal and Learning Control

for

Autonomous Robots

Lecture 1

ADRL

Jonas Buchli
Agile & Dexterous Robotics Lab

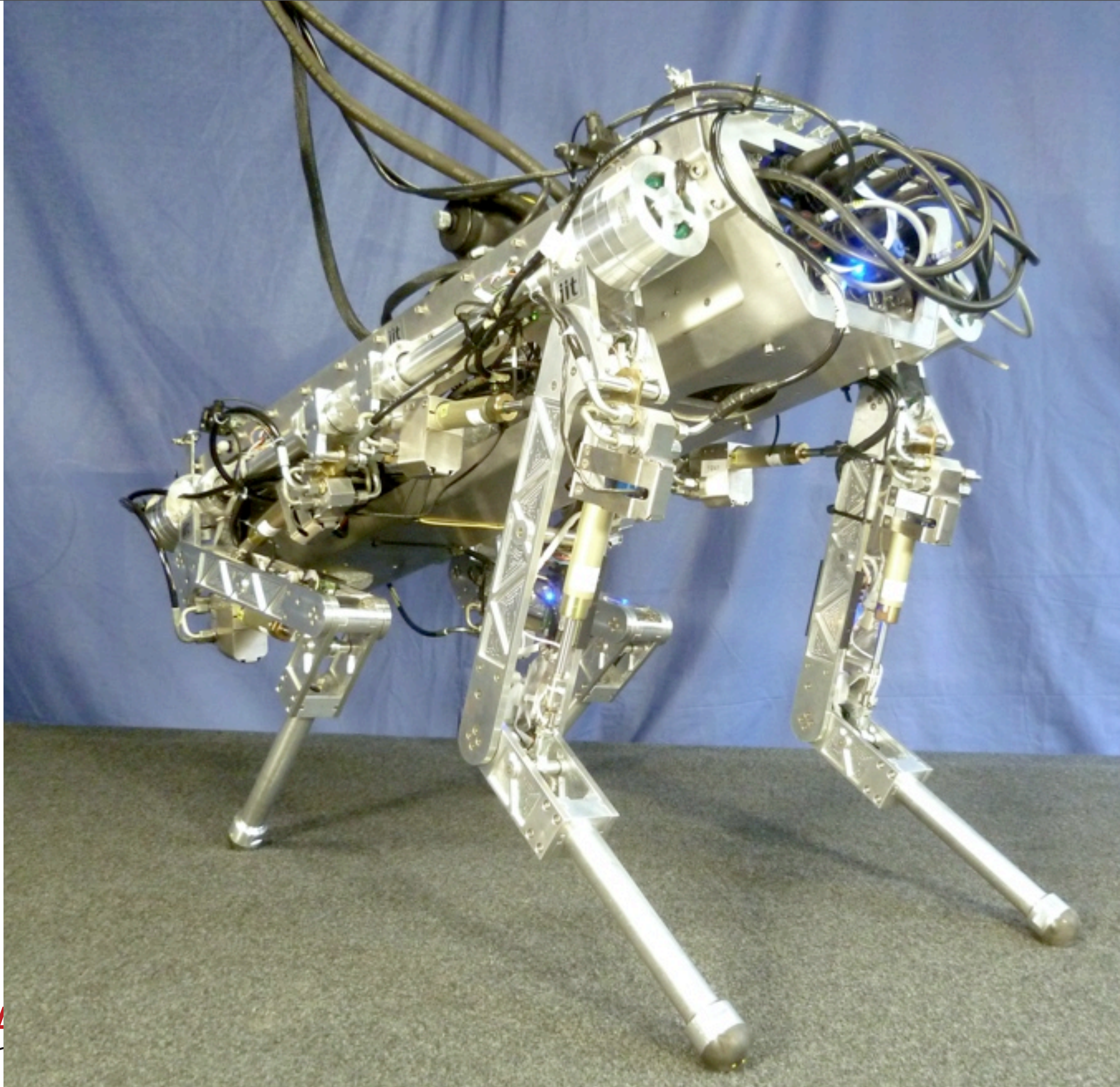
FNSNF

ETH Zürich

Lecture 1

- Intro
- Adminstrativa and Logistics
- Reinforcement learning
- some important concepts and terms
- Modeling / math. prerequisites
- Examples

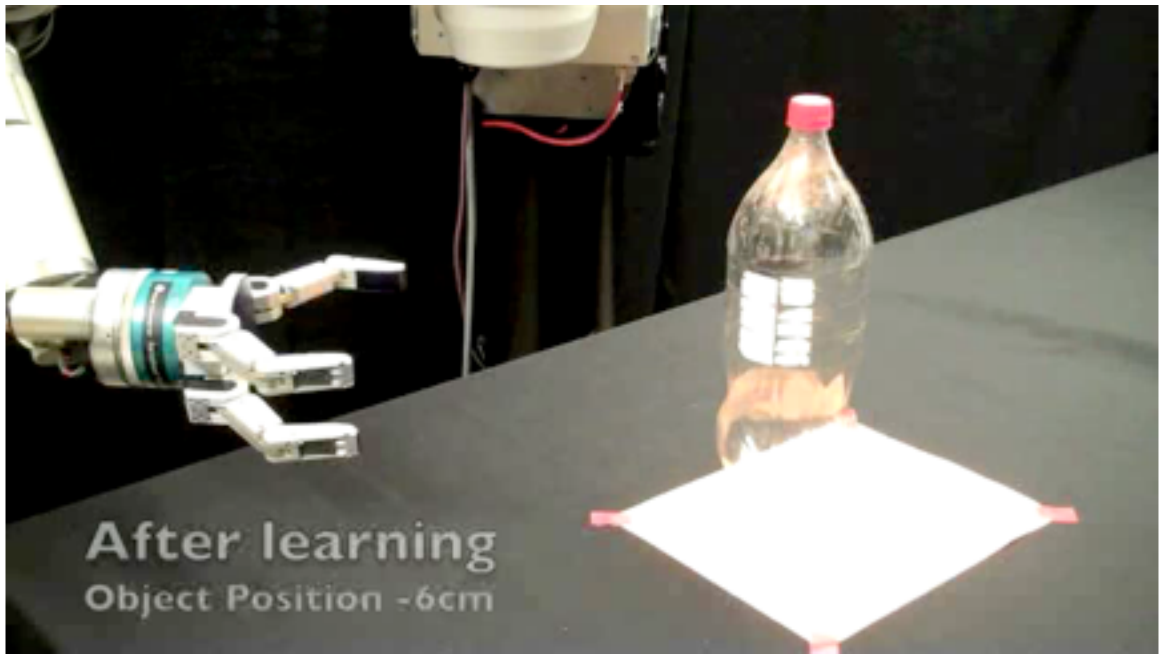
L1 Reading material:
Script Ch 1.1

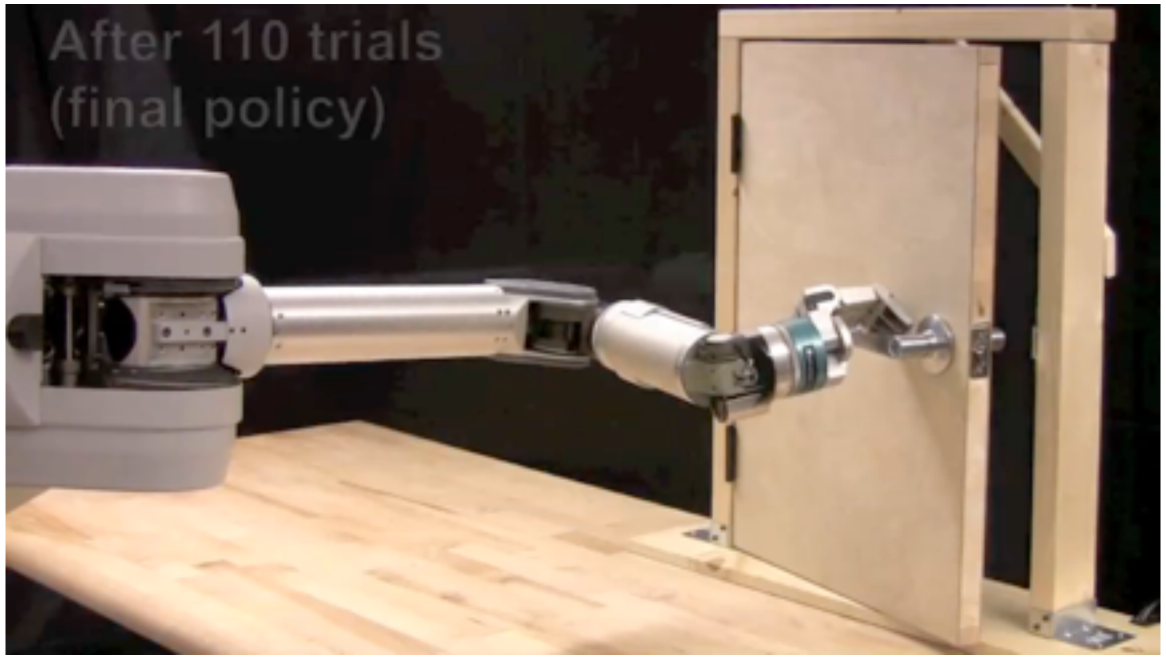


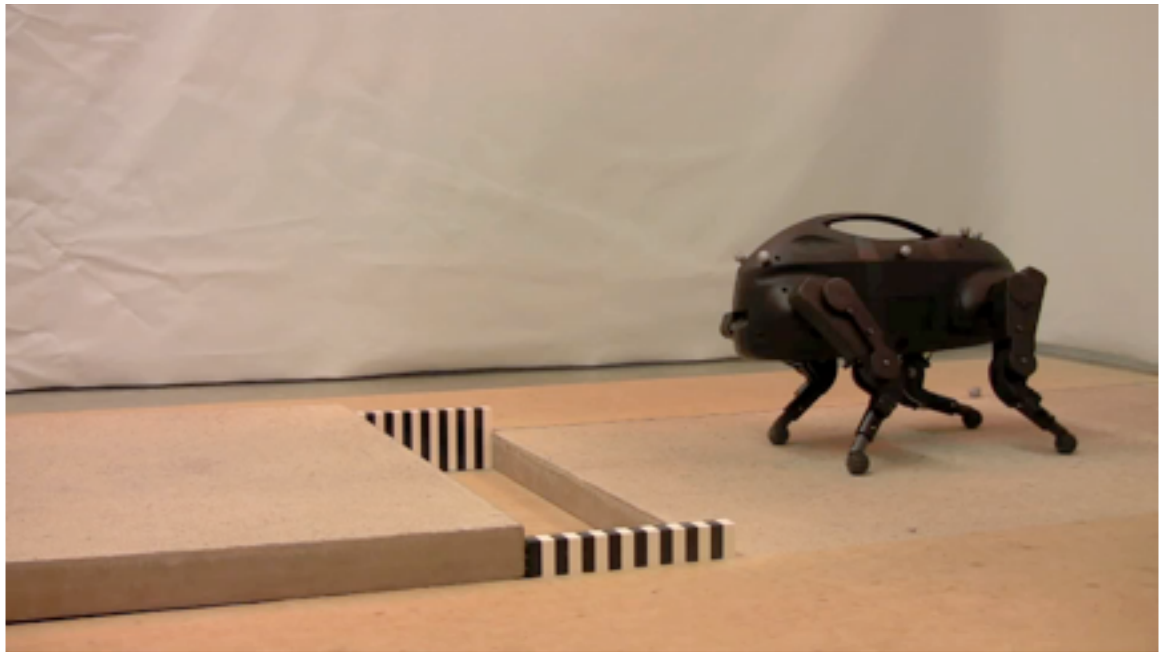
TH Zürich

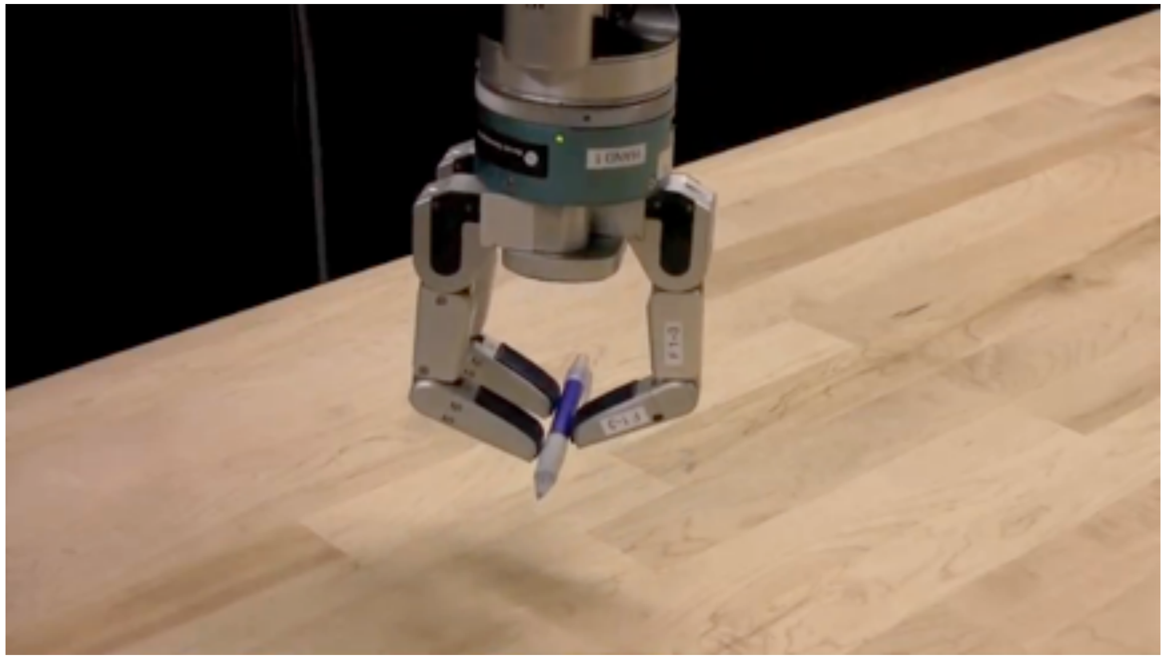
Learning Complex Movement Skills

CLMC Lab



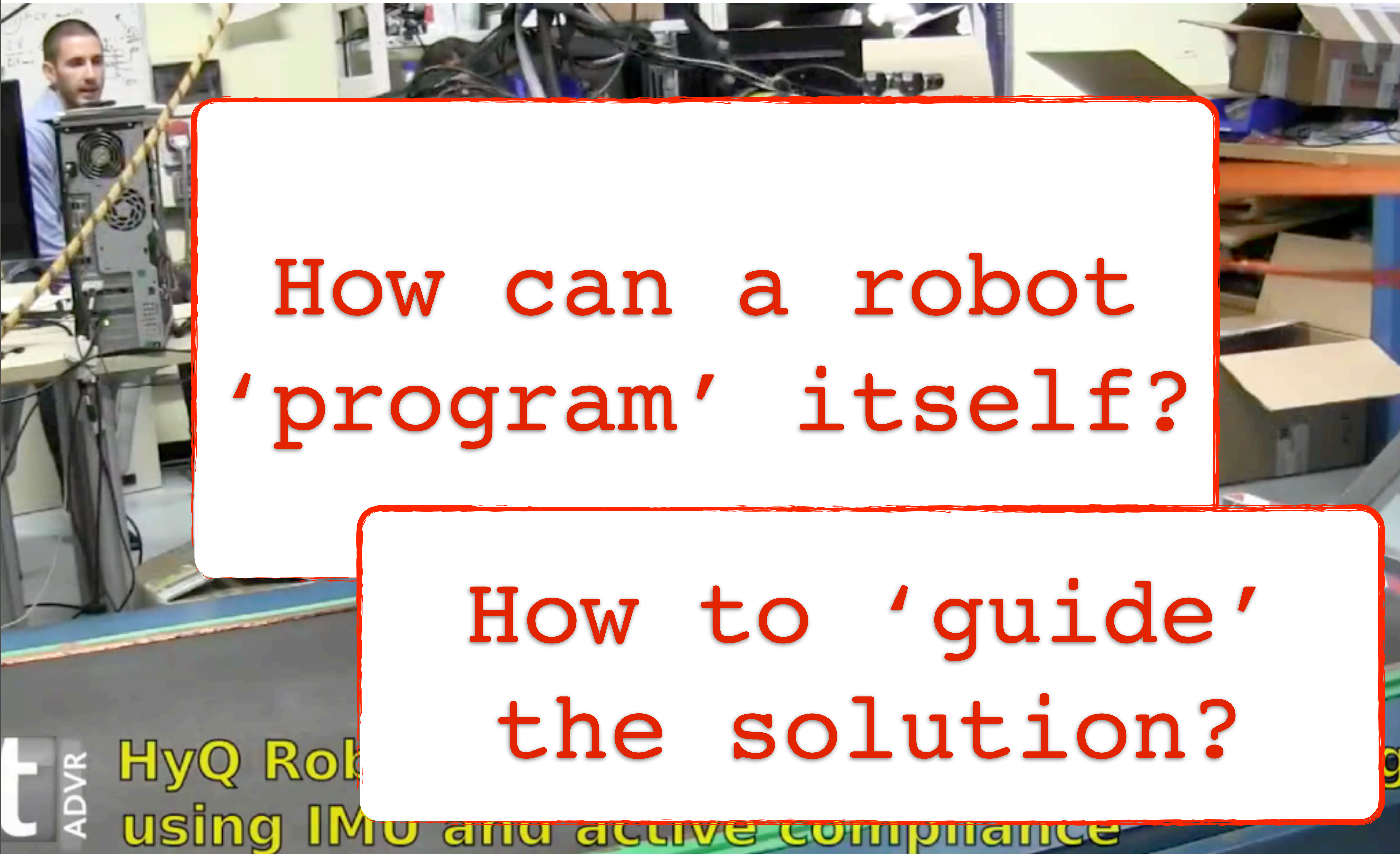






ADRL

ETH Zürich

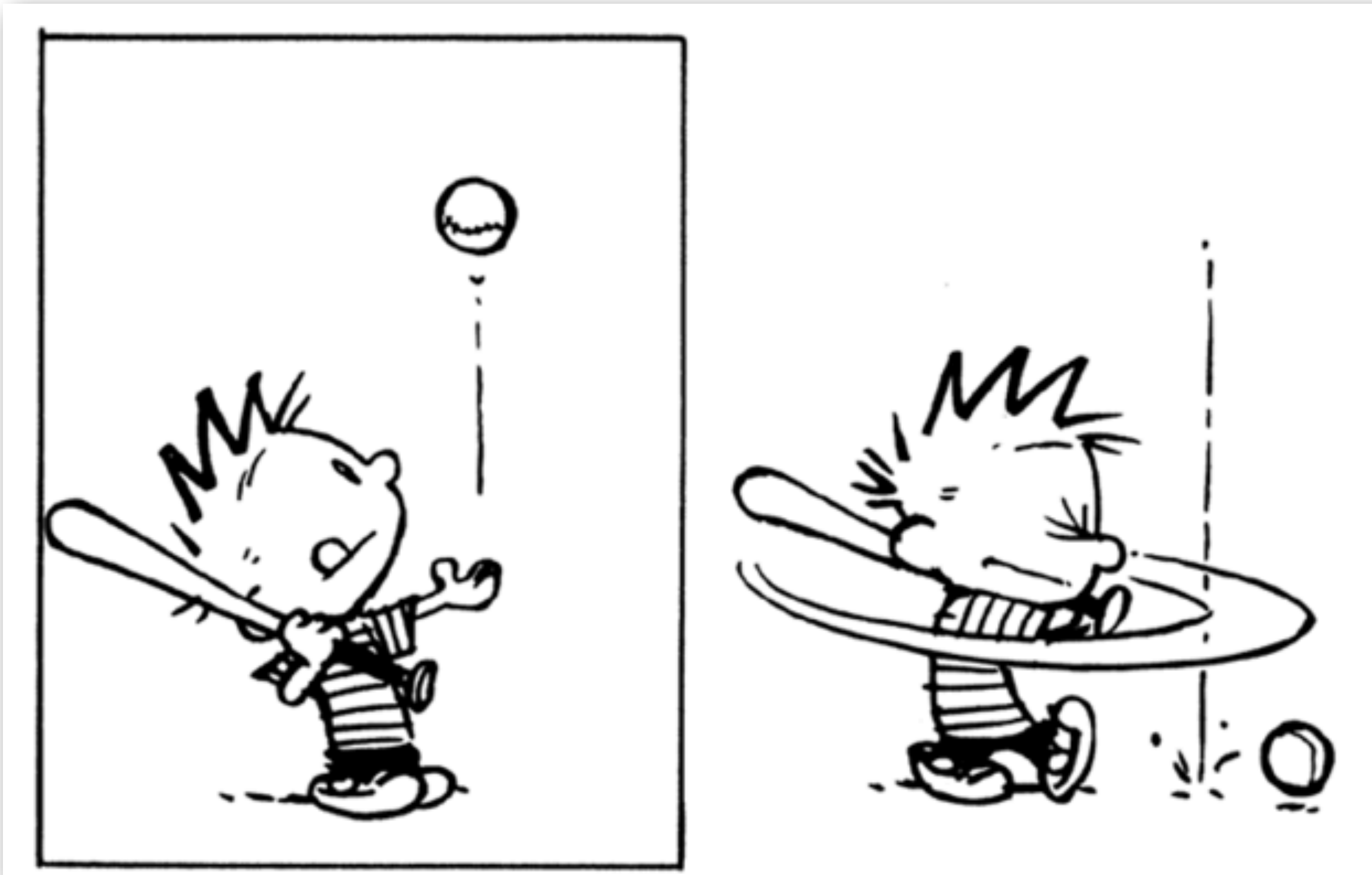


How can a robot 'program' itself?

How to 'guide' the solution?

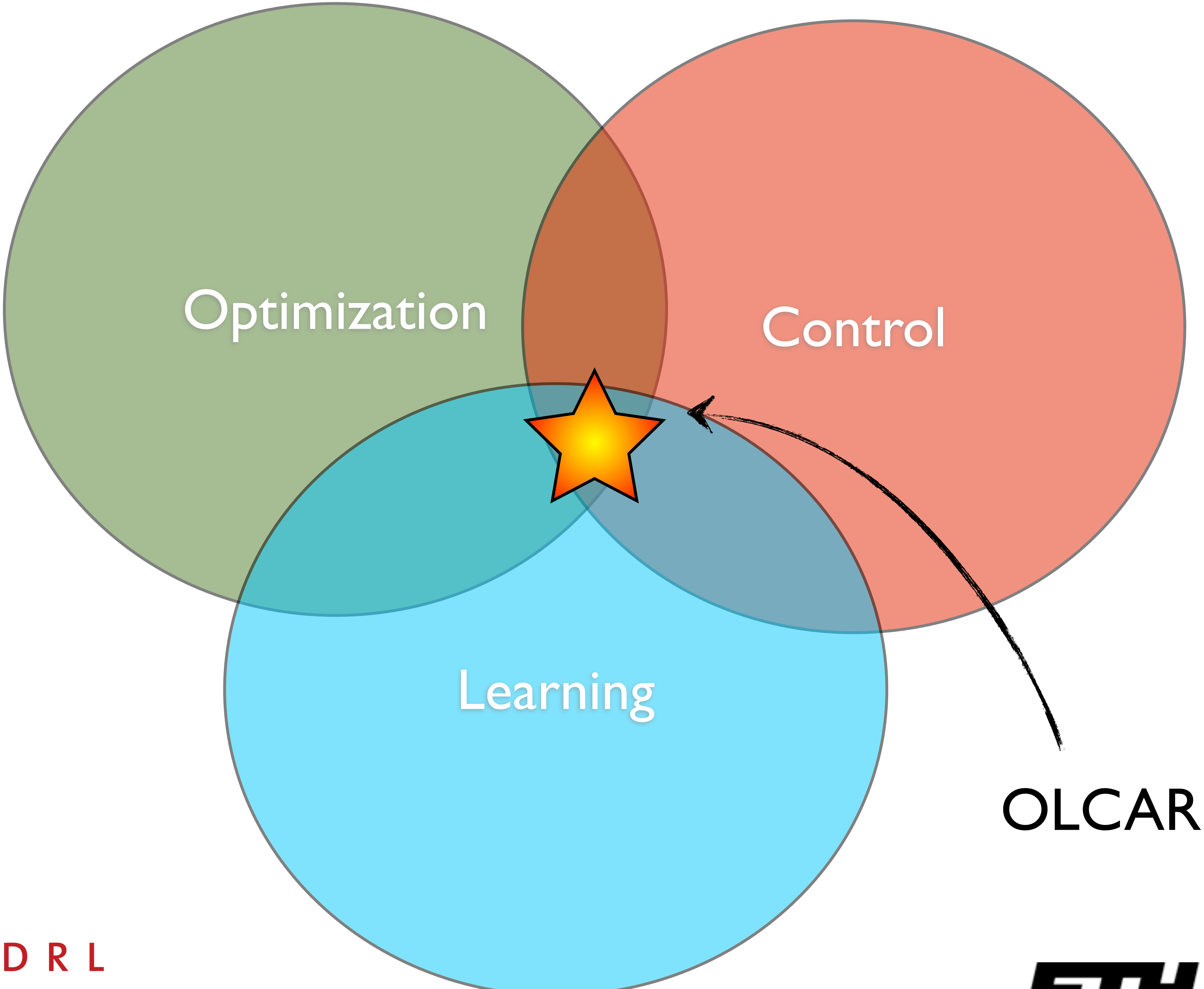
A D R L

Optimal and learning control



★Formalizing the problem of 'how to do things well'

★Derive 'ways to do things well'



A D R L

ETH Zürich

(

Class logistics

Lecturer: Jonas Buchli - buchlij@ethz.ch
Assistant: Farbod Farshidian - farshidian@mavt.ethz.ch

Office hours: Thu, 18-19 Room: TBA
(no office hours this week)

Website:
http://www.adrl.ethz.ch/doku.php/adrl:education:lecture:fs2015

Language: English

Exercises/Exam

Exercises

- 3 programming exercises
- starting L5, 8, 12
- exercises graded pass/fail
- grade boost for passed exercises
  - ➡Ex1: 0.1, Ex 2: 0.05, Ex 3: 0.1
- solutions will be available at end of semester
- topics of exercises will be used for exam

Exam: written, english, 2h

more details TBA

Prerequisites

Required

Motivation & Interest!

Programming
Basic systems theory
Basic control theory
Basic Calculus
Basic Functional Analysis
Probability

Beneficial

Optimal Control
Dynamic Programming

$$J = E\left\{\Phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} L(\mathbf{x}(t'), \mathbf{u}(t'))dt'\right\}$$

$$\begin{aligned}\Delta V^*(t,x) &\approx \frac{dV^*(t,x)}{dt}\Delta t \\ &= E\left\{\frac{\partial V^*(t,x)}{\partial t}\Delta t + \left(\frac{\partial V^*(t,x)}{\partial x}\right)^T \dot{x}\Delta t + \frac{1}{2}\dot{x}^T\frac{\partial^2 V^*(t,x)}{\partial x^2}\dot{x}\Delta t^2\right\}\end{aligned}$$

Outline

Goals

The students will learn the fundamentals of optimal and learning control. They will learn how these fundamental ideas can be applied to real world problems encountered in autonomous and articulated robots.

After this lecture the students will have the understanding and tools to apply learning and optimal control to problems encountered in robotics and other fields.

Relationship between Optimal Control and Learning

Lecture	Syllabus	Sections
Lecture 1	Introduction, Problem Definition, Principle of optimality,	1.1
Lecture 2	Finite/ Infinite time horizon Bellman equation	1.2
Lecture 3	Finite/ Infinite time horizon HJB equation	1.3
Lecture 4	Iterative Algorithm SQP & SLQ, Motivating for robotic platform	1.5
Lecture 5	ILQC	1.6
Lecture 6	LQR/ LQG continuous and discrete time	1.7 & 1.8
Lecture 7	MDP, Policy evaluation, Value Iteration	
Lecture 8	Monte Carlo, Q-Learning	
Lecture 9	Path Integral, Function approximation	
Lecture 10	PI2 – for trajectory optimization	
Lecture 11	PI2 – for motion control optimization, Variable impedance learning	
Lecture 12	The Framework of motion control – form model-based to sample-based	
Lecture 13	Policy Gradient, Finite difference	
Lecture 14	Summary and Exam Discussion	

Subject to change!

A D R L

ETH Zürich

Literature/Script

Script hardcopy will be given.

Books:

Stengel, Optimal Control and Estimation
http://www.amazon.com/Optimal-Control-Estimation-Dover-Mathematics/dp/0486682005

Bertsekas, Dynamic Programming & Optimal Control
http://www.amazon.com/Dynamic-Programming-Optimal-Control-Vol/dp/1886529264

Sutton & Barto, Reinforcement Learning: An Introduction
http://www.amazon.com/Reinforcement-Learning-Introduction-Adaptive-Computation/dp/0262193981

Papers - will be on website

Feedback appreciated!

)

Lecture 1 Goals

★Have heard the most important terminology of optimal control

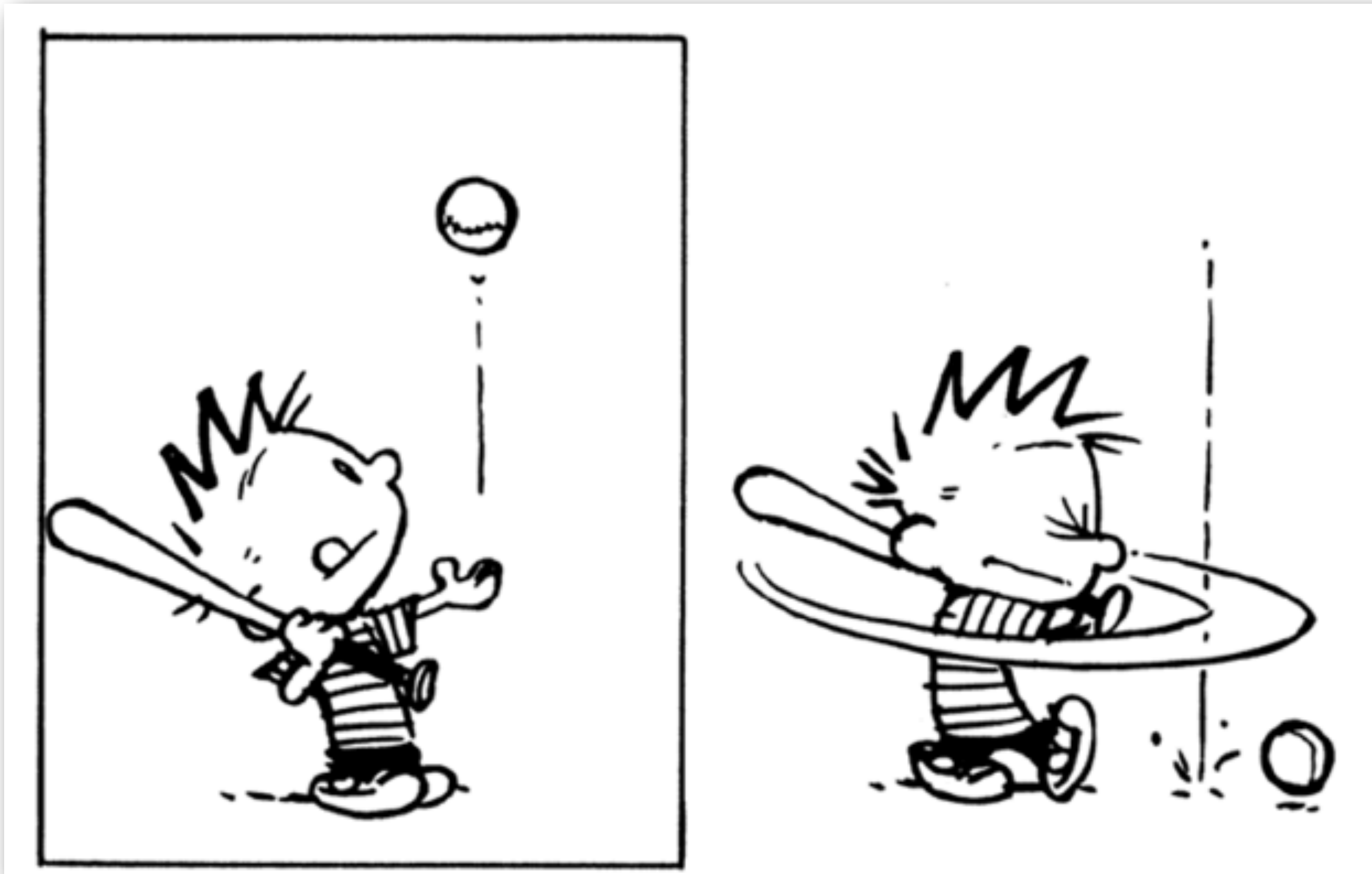
★Understand the scalar nature of cost and reward

★Understand optimization as function minimization

★Understand the principle of optimality

Optimal and learning control

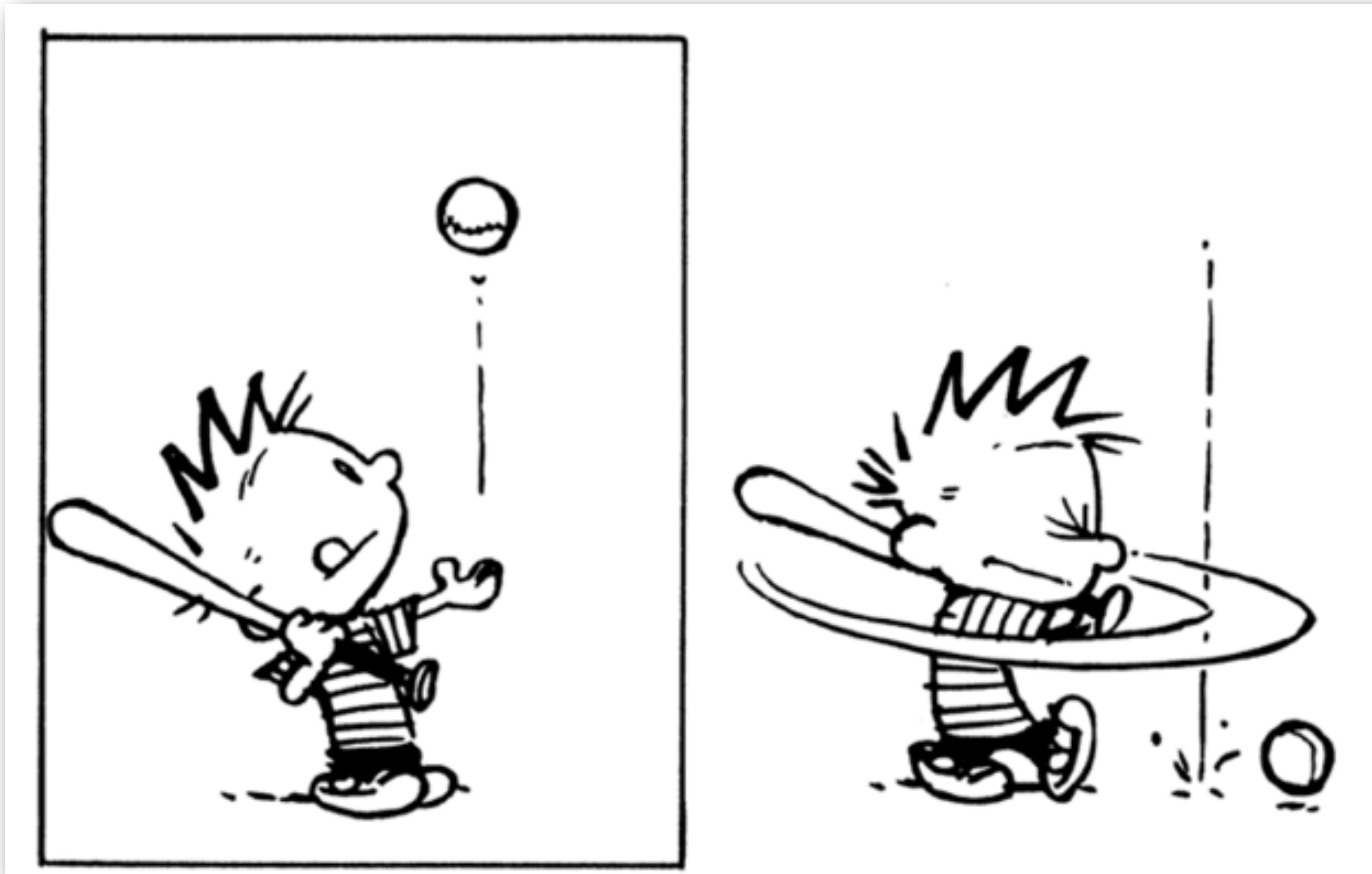
Optimal and learning control



★Formalizing the problem of 'how to do things well'

★Derive 'ways to do things well'

Reinforcement Learning



Learning from unspecific reward
'by trial and error' - delayed reward

Formalize learning problem

Need to formalize this problem...
ideally on a ‘high’ level...

Reinforcement learning terminology

- ★Reward (cost)
  What is good?

Objective: optimize reward

- ★Policy / Controller
  What do do?
- ★Value function
  Which states are good (potential reward!)
- ★System dynamics / Model
  What happens ‘next’?

Can be probabilistic

The goal hitting a baseball is....

$$R = \text{distance of ball}$$

Cost and reward functions

‘A single number describing the quality of the solution’

Quality dependent on some parameters

- ☞ Simple example: design a tank, use minimum amount of material
- ☞ Complicated example: Minimize boarding time of a plane

Example - ‘Static’ optimization

$\Leftrightarrow$ parameter optimization

How to solve for optimum?

- ✓ Analytically
- ✓ Numerically using a model:
  Root finding/Gradient descent
- ✓ Sample real world:
  Numerically process ‘experience’
  Explicit gradient / implicit gradient

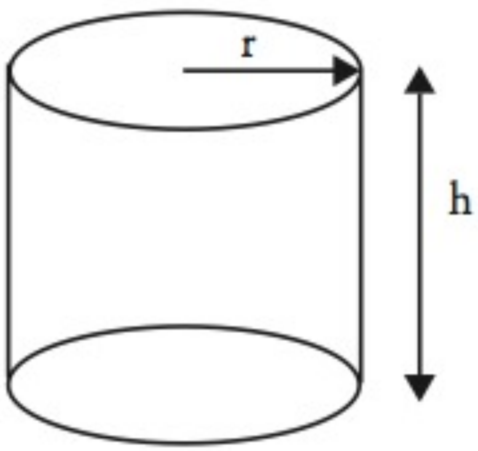
NOTE - to be more coherent with the other introduced notation, the x in the following example should be replaced by θ (parameters of the policy, i.e. the x in this example is not to be confused with the states).

Example: build a tank

Use minimum material, for a given Volume

V=10

$$V = \pi r^2 h$$



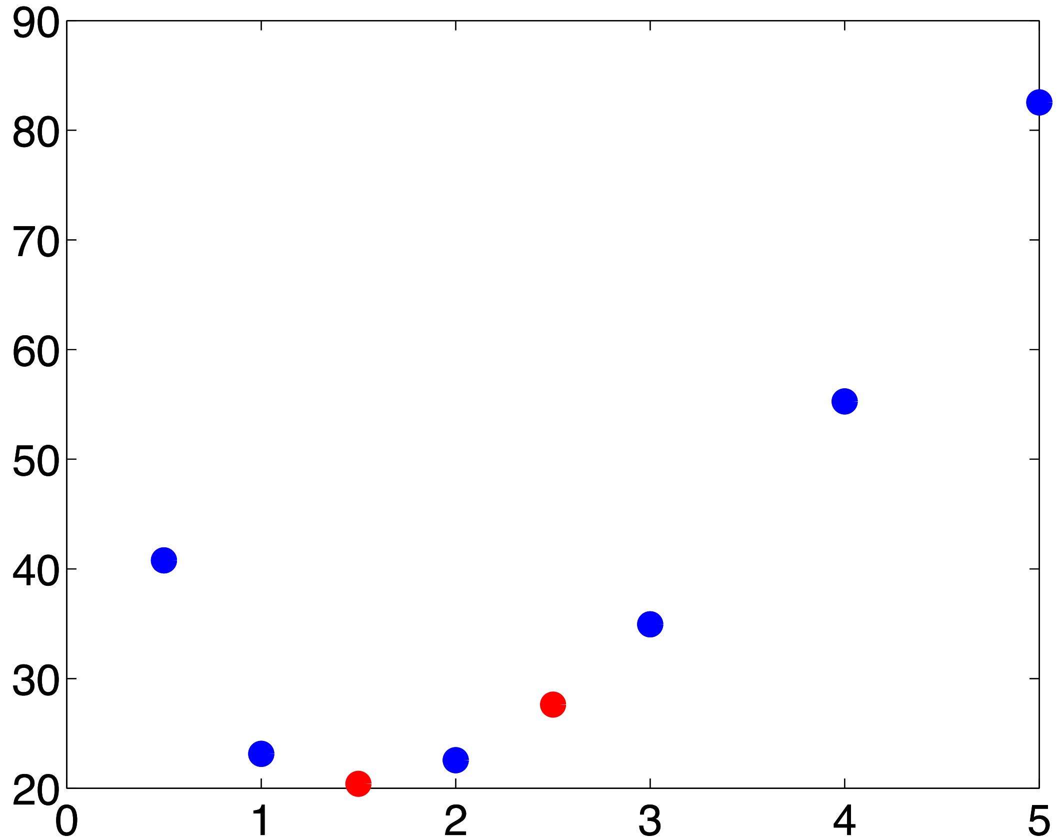
Model! Sampled...

Cost: $A = \pi r^2 + 2\pi r h$

$$h = \frac{V}{\pi r^2} \Rightarrow A = \pi r^2 + \frac{2V}{r}$$

r	A
0.5	40.79
1	23.14
2	22.56
3	34.94
4	55.27
5	82.54

1.5	20.40
2.5	27.64



r_opt is approx. 1.5

A D R L

Naive approach: brute force search

ETH Zürich

How to solve for optimum?

✓Analytically

✓Numerically using a model:
Root finding/Gradient descent

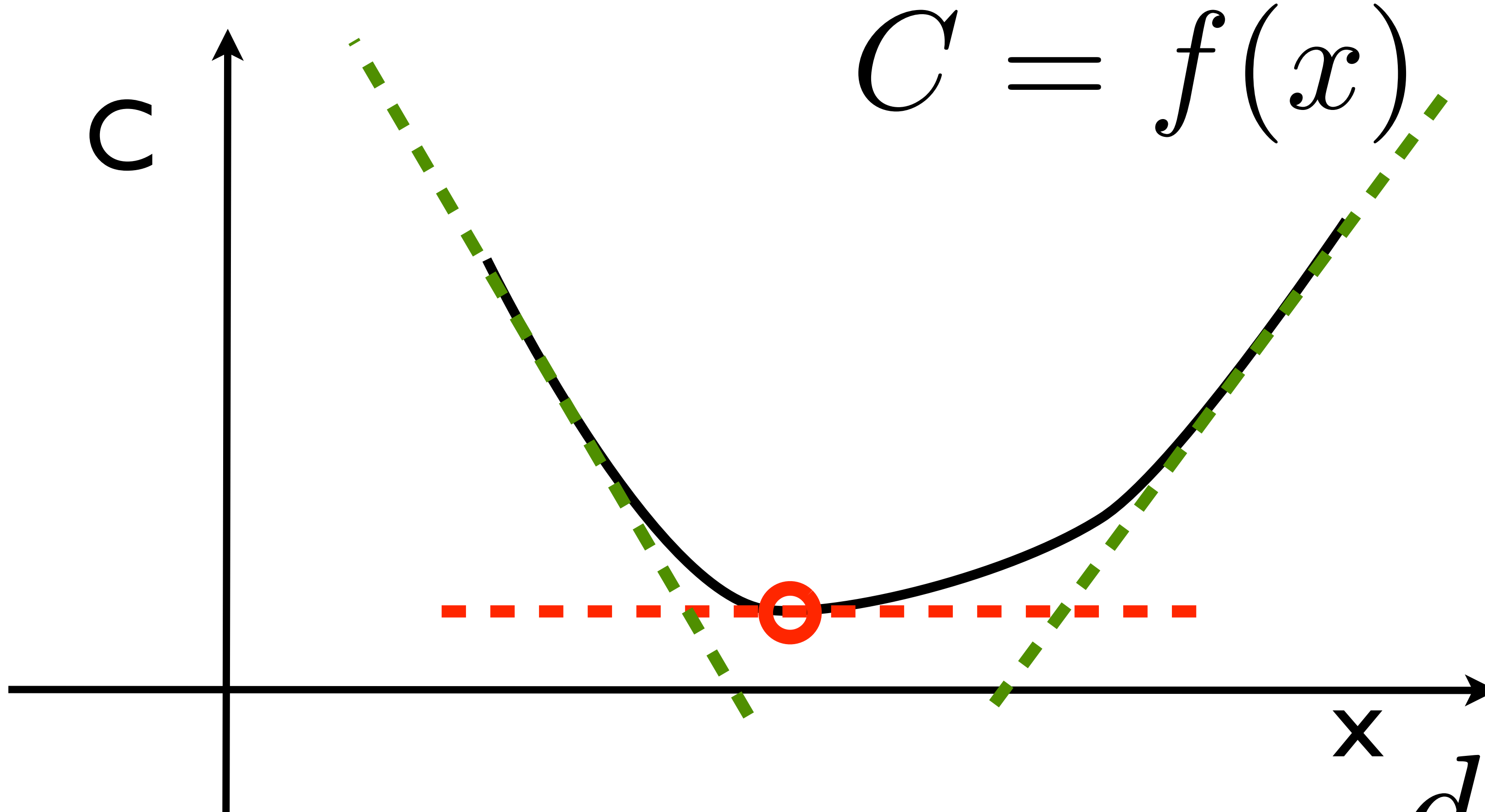
✓Sample real world:
Numerically process ‘experience’
Explicit gradient / implicit gradient

Analytical optimum

Minima (and maxima) of functions

1 dimensional:

$$C = f(x)$$

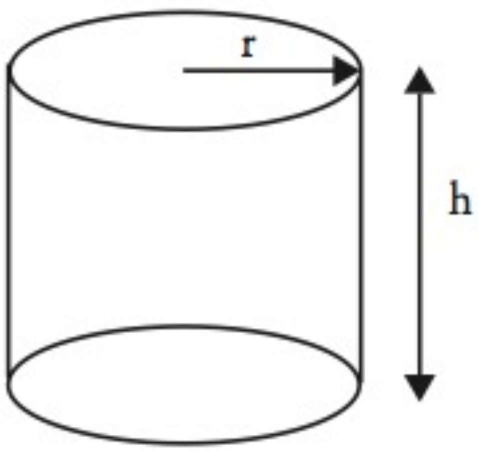


$$\frac{dC}{dx} = 0$$

Minimum is an 'inflection point' - slope is 0

Example: build a tank

Use minimum material, for a given Volume

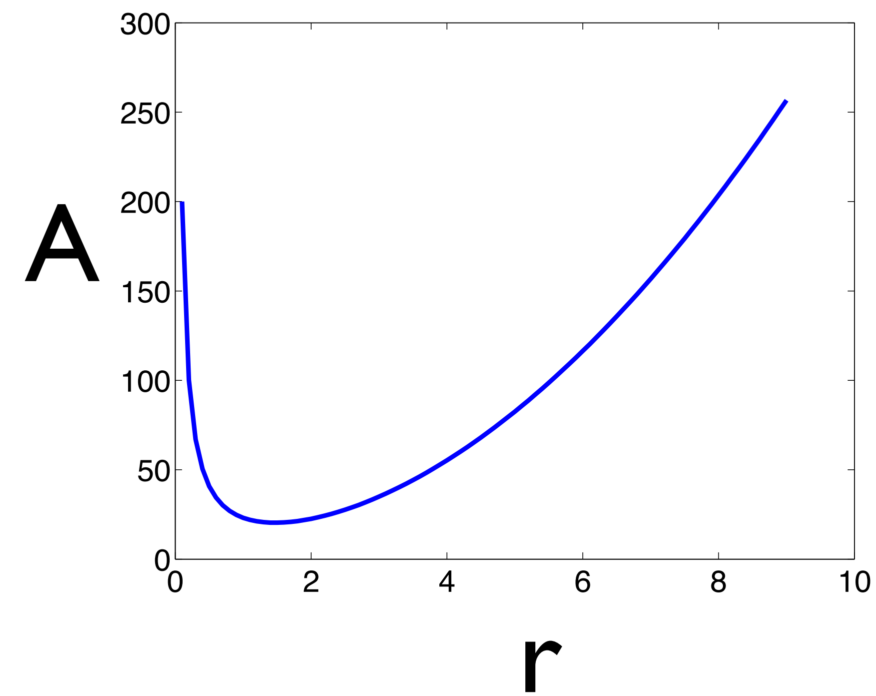


$$V = \pi r^2 h$$

Cost: $A = \pi r^2 + 2\pi r h$

$$h = \frac{V}{\pi r^2}$$

$$A = \pi r^2 + \frac{2V}{r}$$



Parameter: r

$$\frac{\partial A}{\partial r} = 2\pi r - \frac{2V}{r^2} = 0$$

$$2\pi r^3 - 2V = 0 \quad \Rightarrow \quad r^3 = \frac{2V}{2\pi}$$

$$\Rightarrow \quad V = 10 \rightarrow r = 1.4710$$

Root finding

Example 1: Using analytical gradient, find roots

$$C(x) = a(x - x_0)^2 + b(x - x_1) + c$$

$$\frac{\partial C(x)}{\partial x} = 2a(x - x_0) + b$$

$$x_{opt} = x_0 - \frac{b}{2a}$$

Example 1.1: 4th order polynomial

$$C(x) = ax^4 + bx^3 + cx^2 + dx + e$$

$$\frac{\partial C(x)}{\partial x} = 4ax^3 + 3bx^2 + cx + d$$

Example 1.2: nth order polynomial

$$C(x) = \sum_{i=0}^{n} w_i x^i$$

$$\frac{\partial C(x)}{\partial x} = \sum_{i=1}^{n} i w_i x^{i-1}$$

Use numerical gradient → gradient descent

Analytical optimum

Minima (and maxima) of functions

n-dimensional:

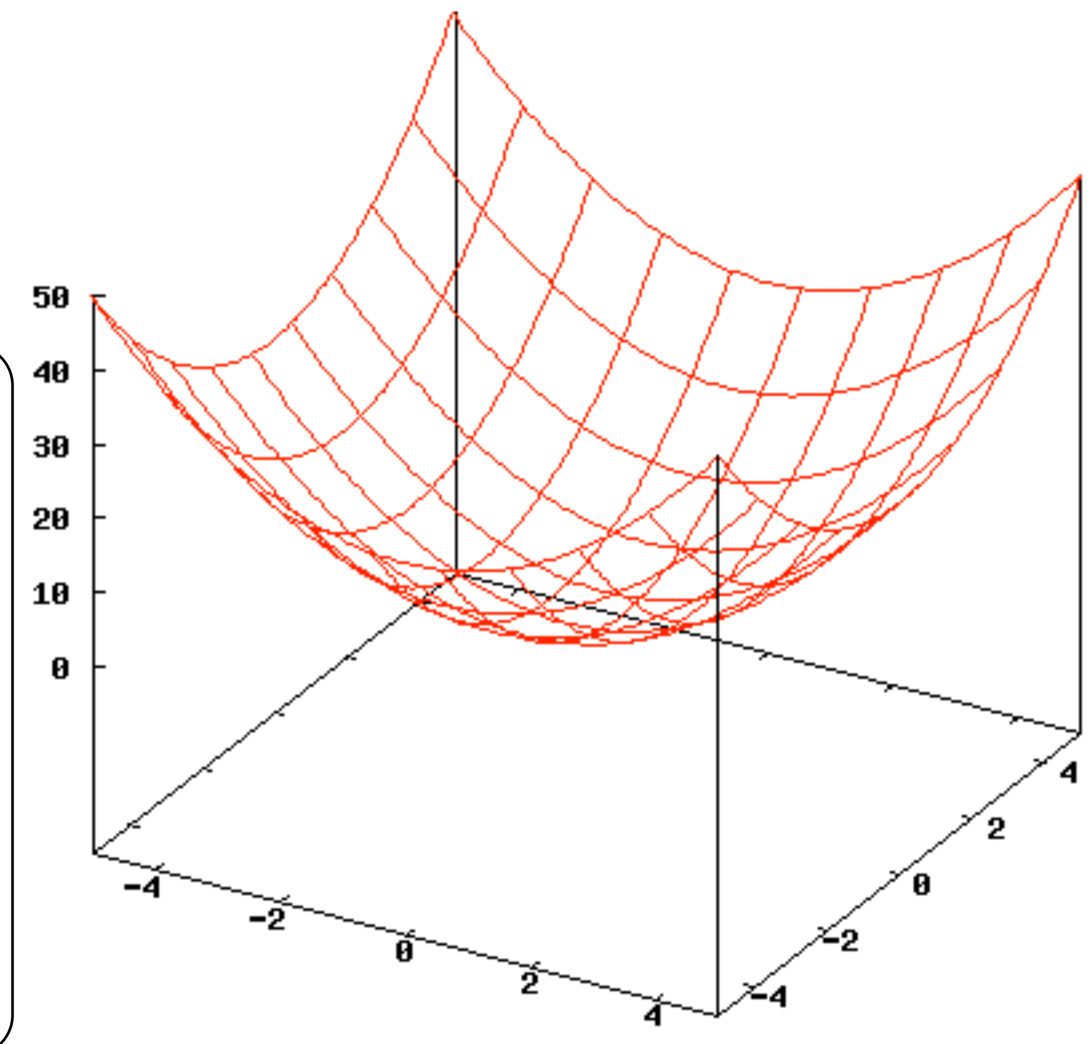
$$C = f(x_1, \ldots, x_n)$$

$$\frac{\partial C}{\partial x_i}$$

$$\nabla C = \left[\frac{\partial C}{\partial x_1}, \ldots, \frac{\partial C}{\partial x_n}\right]^T$$

$$\frac{\partial C}{\partial x_i} = 0$$

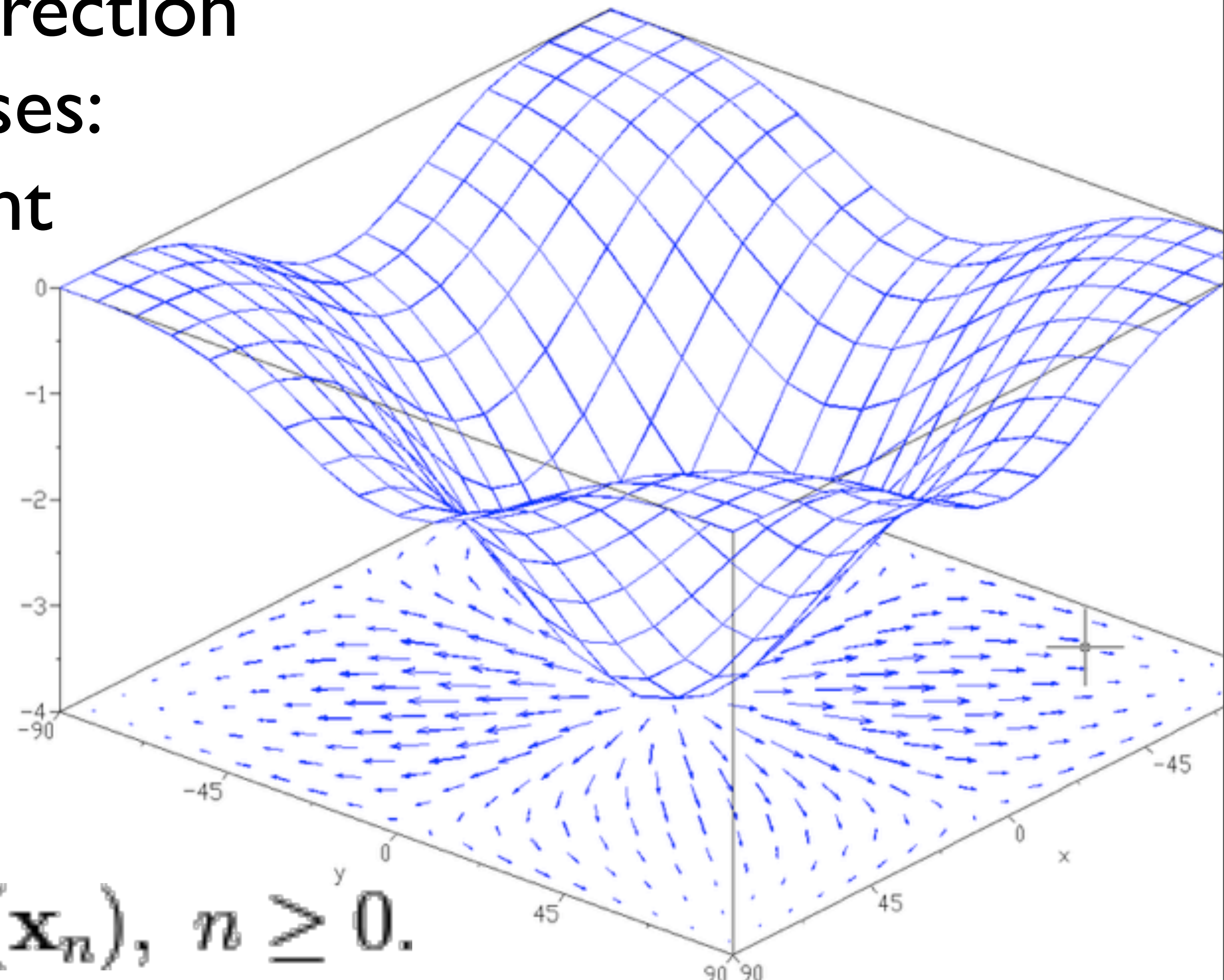
$$\nabla C = 0$$



Minimum is an 'inflection point' - slope is 0

Gradient descent

Idea: want to go in direction where C decreases: negative gradient



$$\mathbf{x}_{n+1} = \mathbf{x}_n - \gamma_n \nabla F(\mathbf{x}_n),\ n \geq 0.$$

How to solve for optimum?

✓Analytically

✓Numerically using a model:
Root finding/Gradient descent

Learning

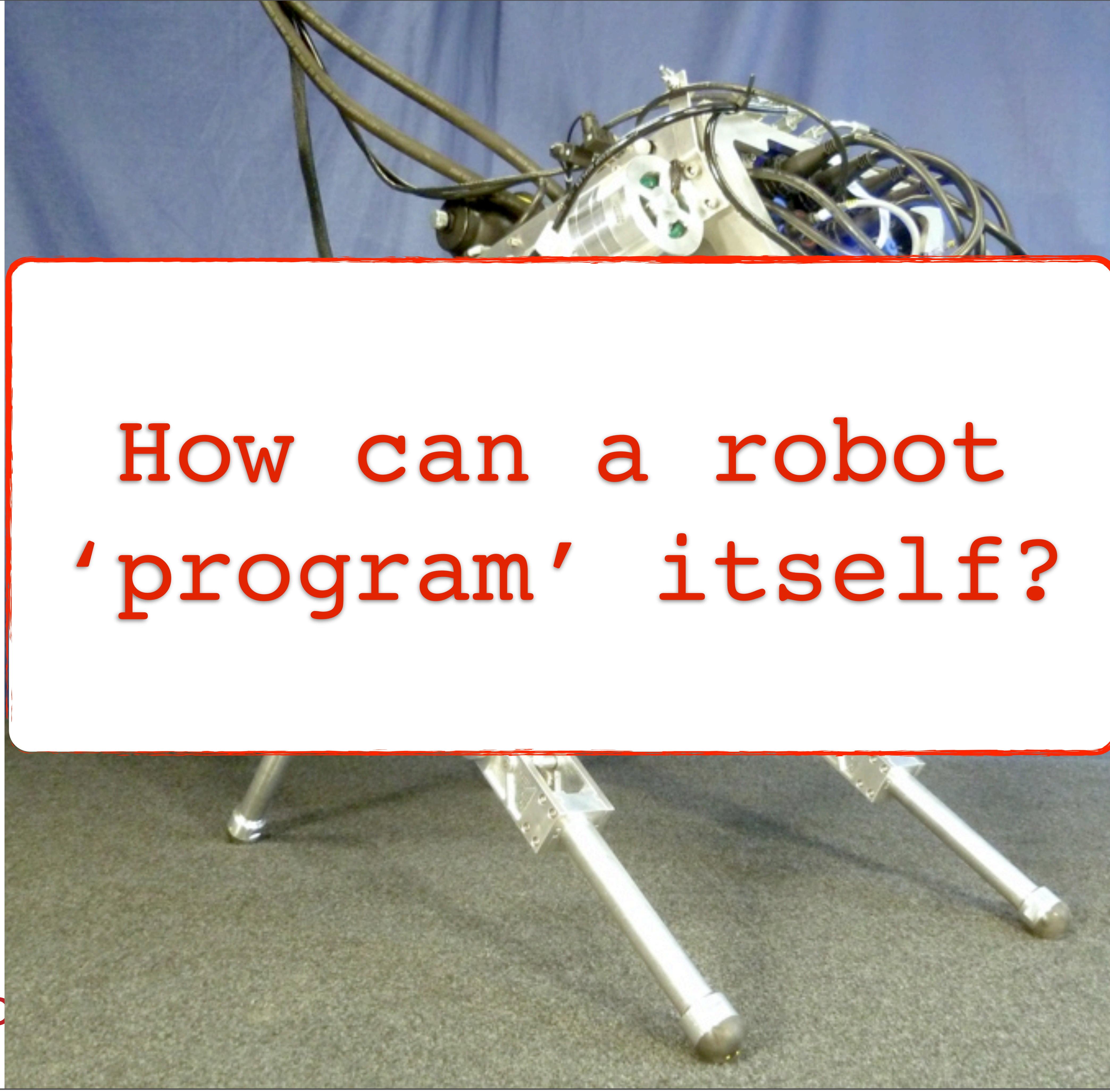
✓Sample real world:
Numerically process ‘experience’
Explicit gradient / implicit gradient

A D R L

ETH Zürich

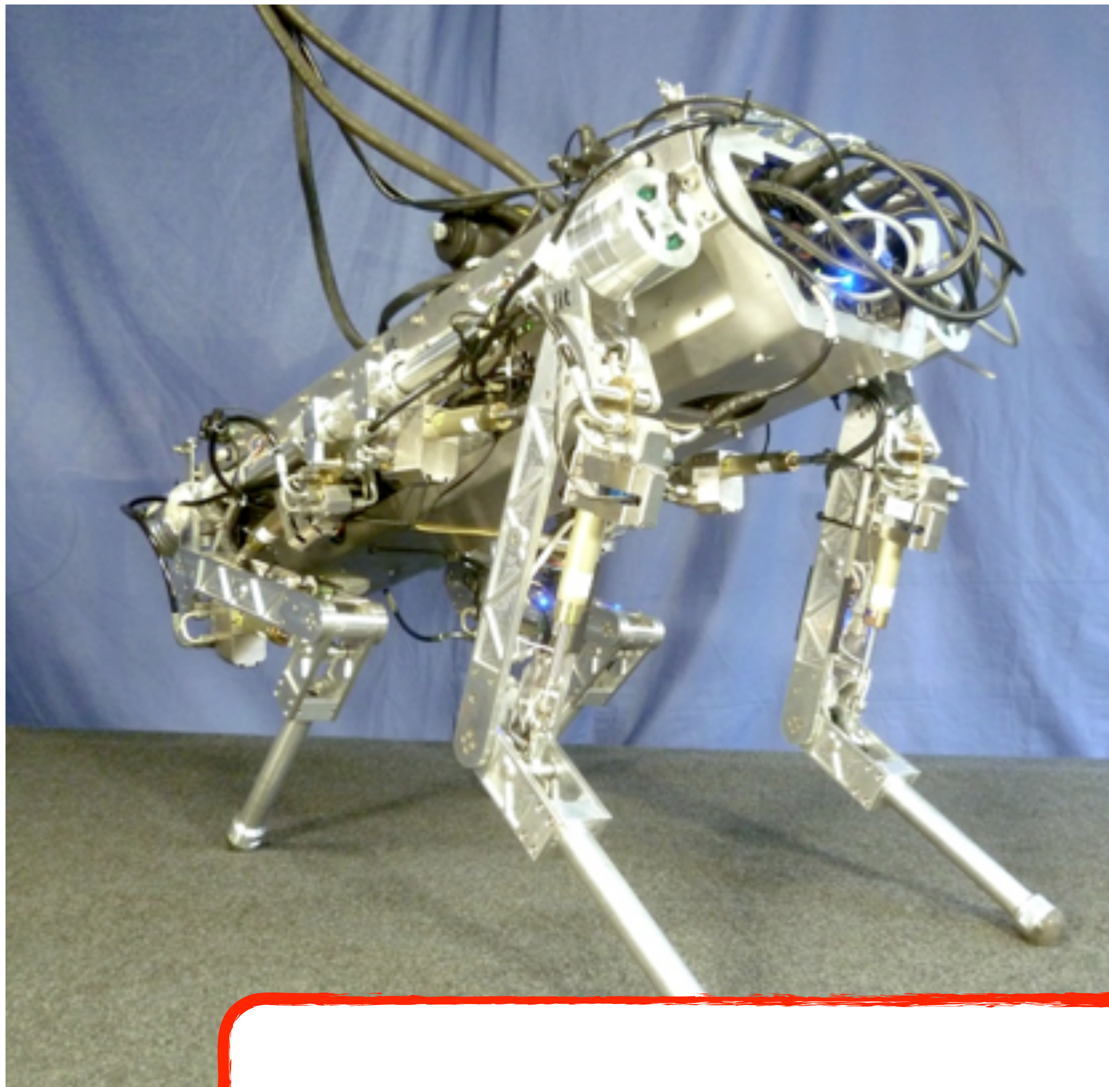
Learning: 'Sampling' the real world, process experience!

[EOF EXAMPLE]



How can a robot ‘program’ itself?

What's a robot?



Models?
Controls?
Open parameters for optimization?

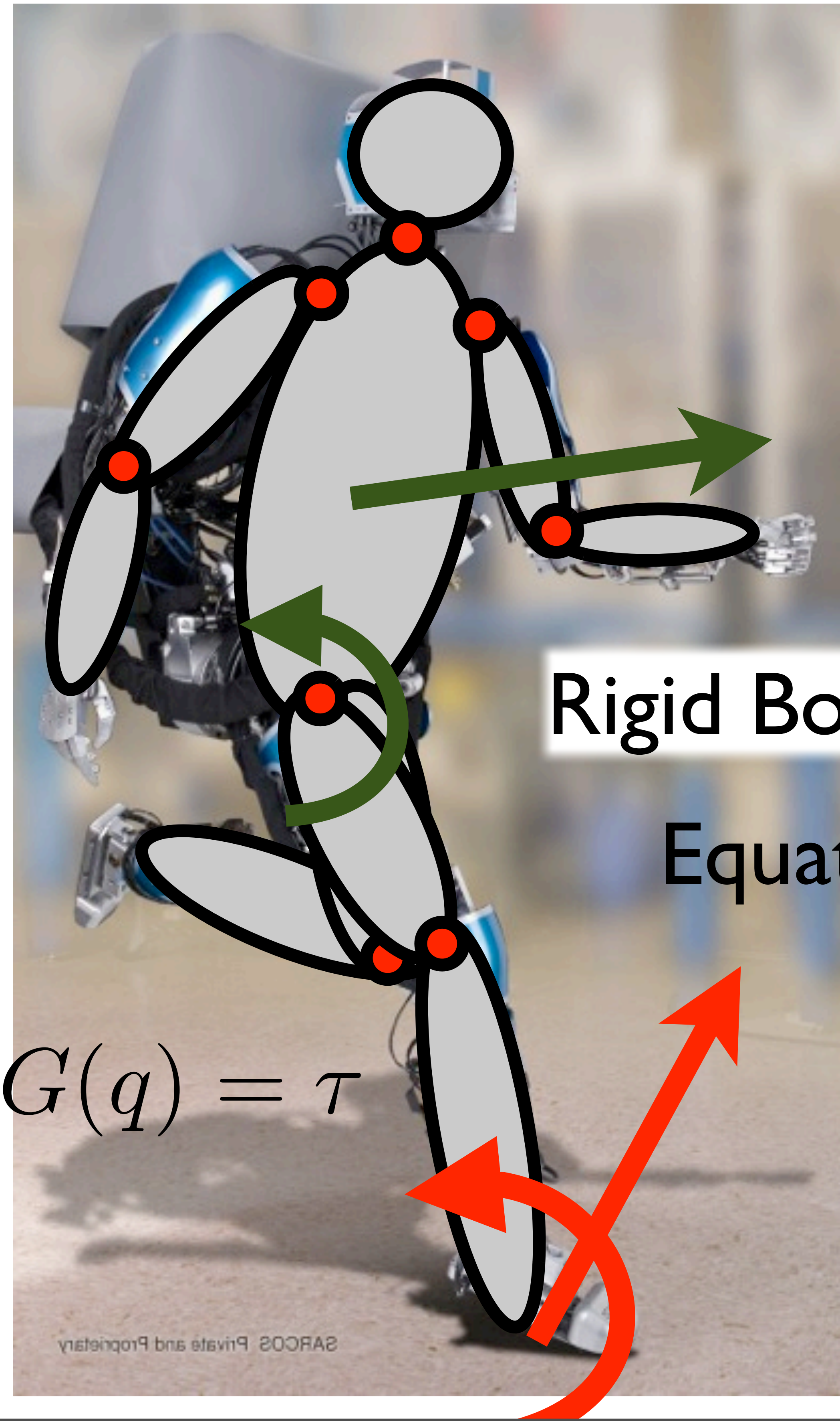
How can a robot 'program' itself?

A D R L

ETH Zürich

Robot model

Rigid body connected by discrete joints



Rigid Body Dynamics?

Equations of motions?

$$M\ddot{q} + C(q,\dot{q}) + G(q) = \tau$$

A D R L

ETH Zürich

Model of a simple robot

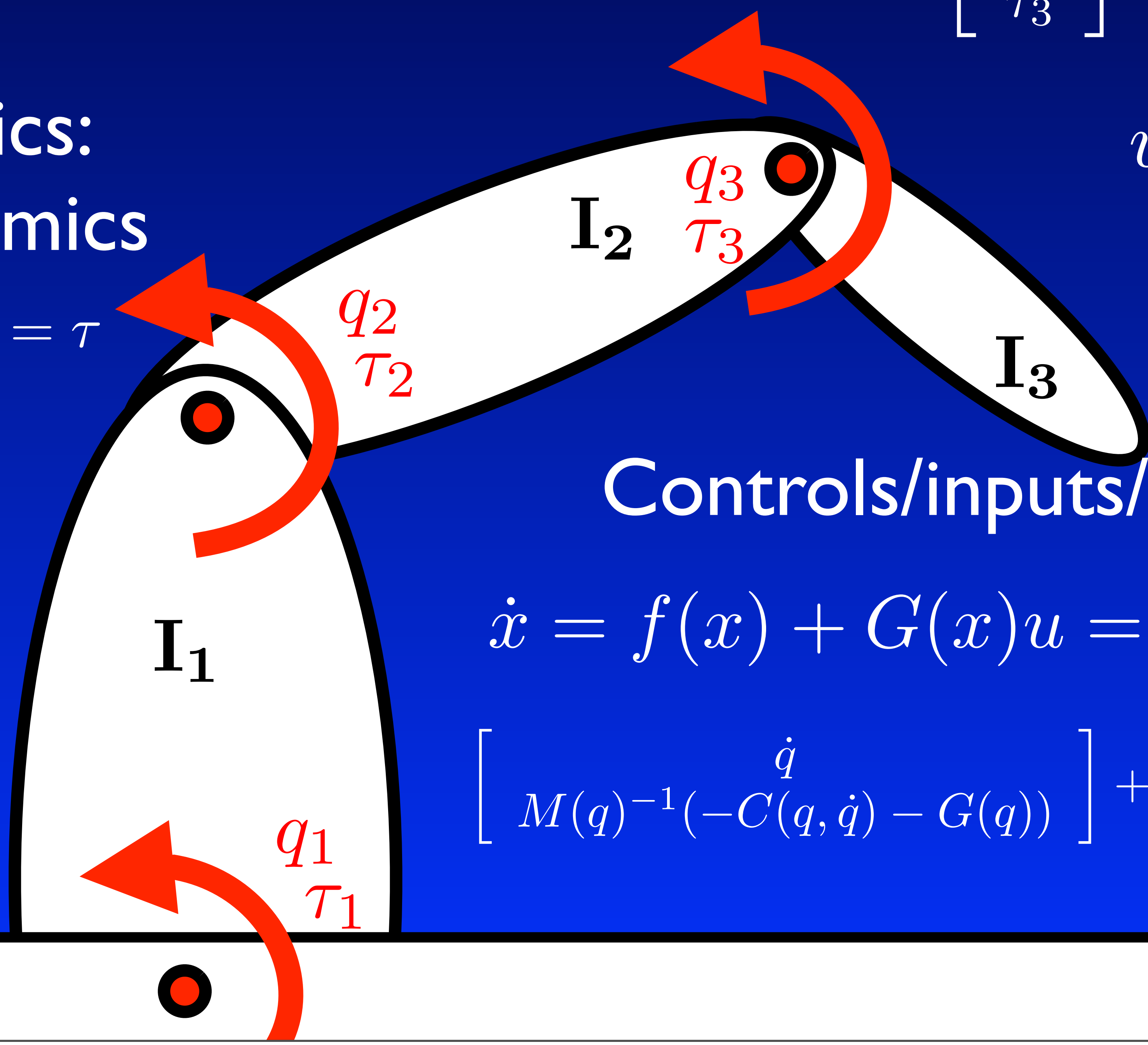
Measurements: Joint angles: $q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$

Actuation: joint torques $\tau = \begin{bmatrix} \tau_1 \\ \tau_2 \\ \tau_3 \end{bmatrix}$

$u = \tau$

Governing physics:
Rigid body dynamics

$$M\ddot{q} + C(q, \dot{q}) + G(q) = \tau$$



System states:

$$x = \begin{bmatrix} q \\ \dot{q} \end{bmatrix}$$

Controls/inputs/actions:

$$\dot{x} = f(x) + G(x)u =$$

$$\begin{bmatrix} \dot{q} \\ M(q)^{-1}(-C(q, \dot{q}) - G(q)) \end{bmatrix} + \begin{bmatrix} 0 \\ M(q)^{-1} \end{bmatrix}^T u$$

Controller/Policy

A mapping from states to actions, possibly dependent on time

actuation = f(states,time) $u = f(x, t)$

$u = f(t)$ 'forward/open loop'

$u = f(x)$ 'state feedback'

$u = Ax$ 'linear state feedback'

$u = Ax + Bc(t)$ 'linear state feedback with external input'

$u = A(t)x$ 'linear time varying state feedback'

Feed forward control

In my shower, I know at which setting of the faucet I get the most comfortable temperature, set faucet to this position immediately

Feedback control

Shower gets hotter, turn down hot water

Planning

Because it's healthy(?!), I use warm water and then cold water → temperature schedule (to be implemented with FF or FB)

Expected 'events'

Unexpected 'events'

Intention

Optimal behavior

Disturbance rejection

vs

Nominal behavior

Planner

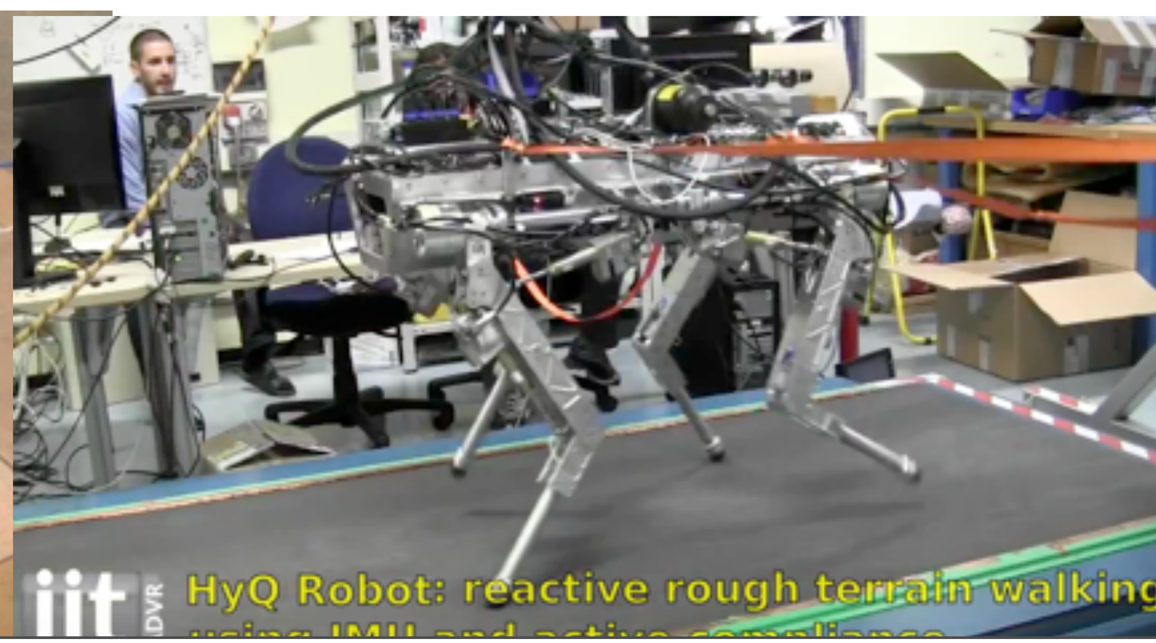
Nominal behavior/performance:

What happens if everything works as assumed

Feedforward controller

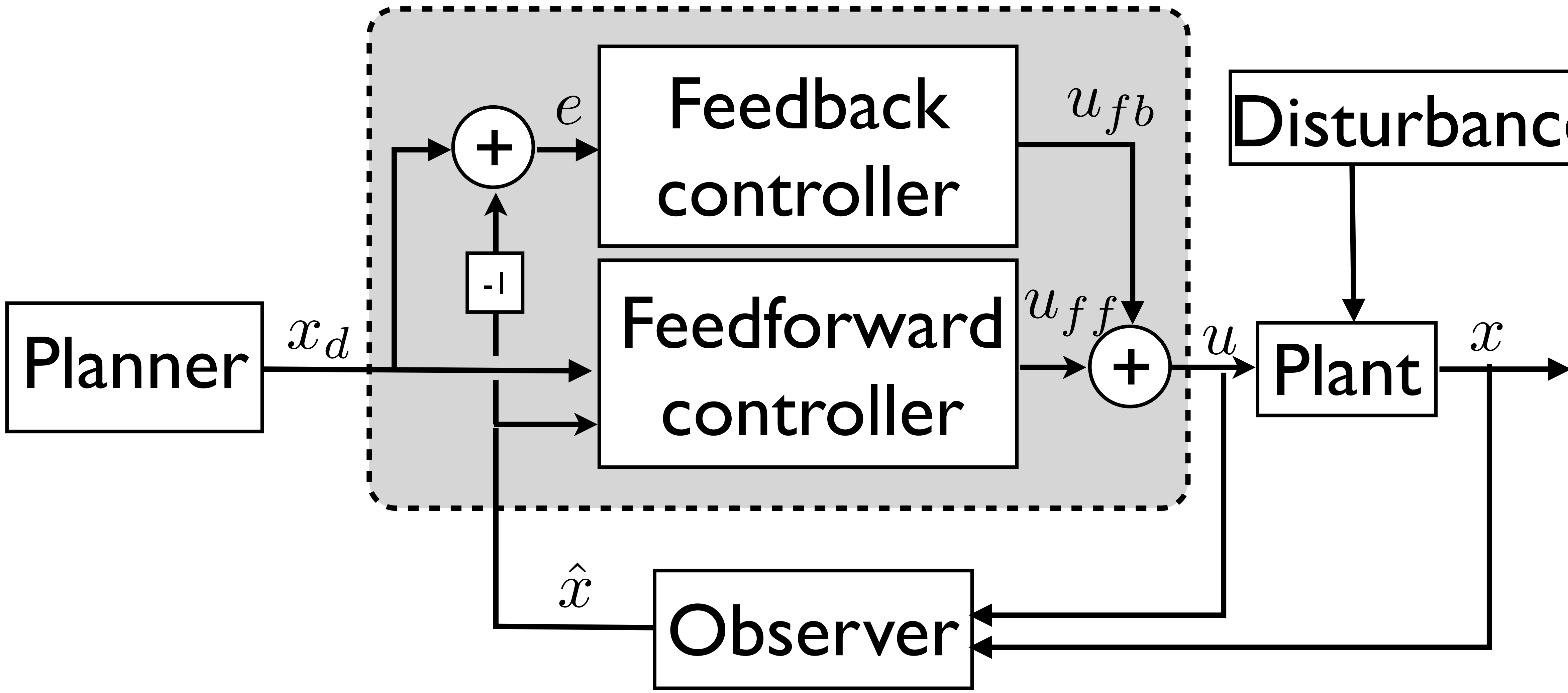
Disturbance rejection/robust performance:

What is the behavior of the system if ‘nominal assumptions’ are violated and/or unexpected things happen



Feedback controller

General control structure



What is a ‘program’?

print “hello world”

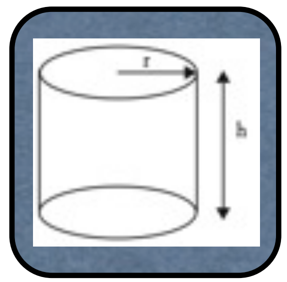
Policies
Controllers

What is a ‘programming’?

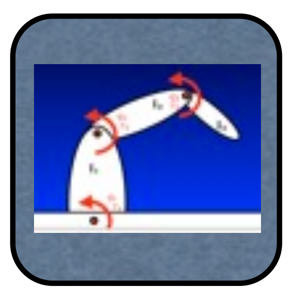
Quadratic program?
Dynamic programming?

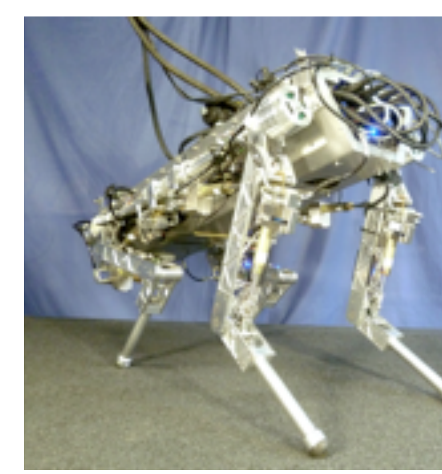
Optimal control?

How to solve for optimum with dynamics affecting the cost?

parameters θ →  → reward $C(X)$

(static) function
$$X = f(\theta)$$

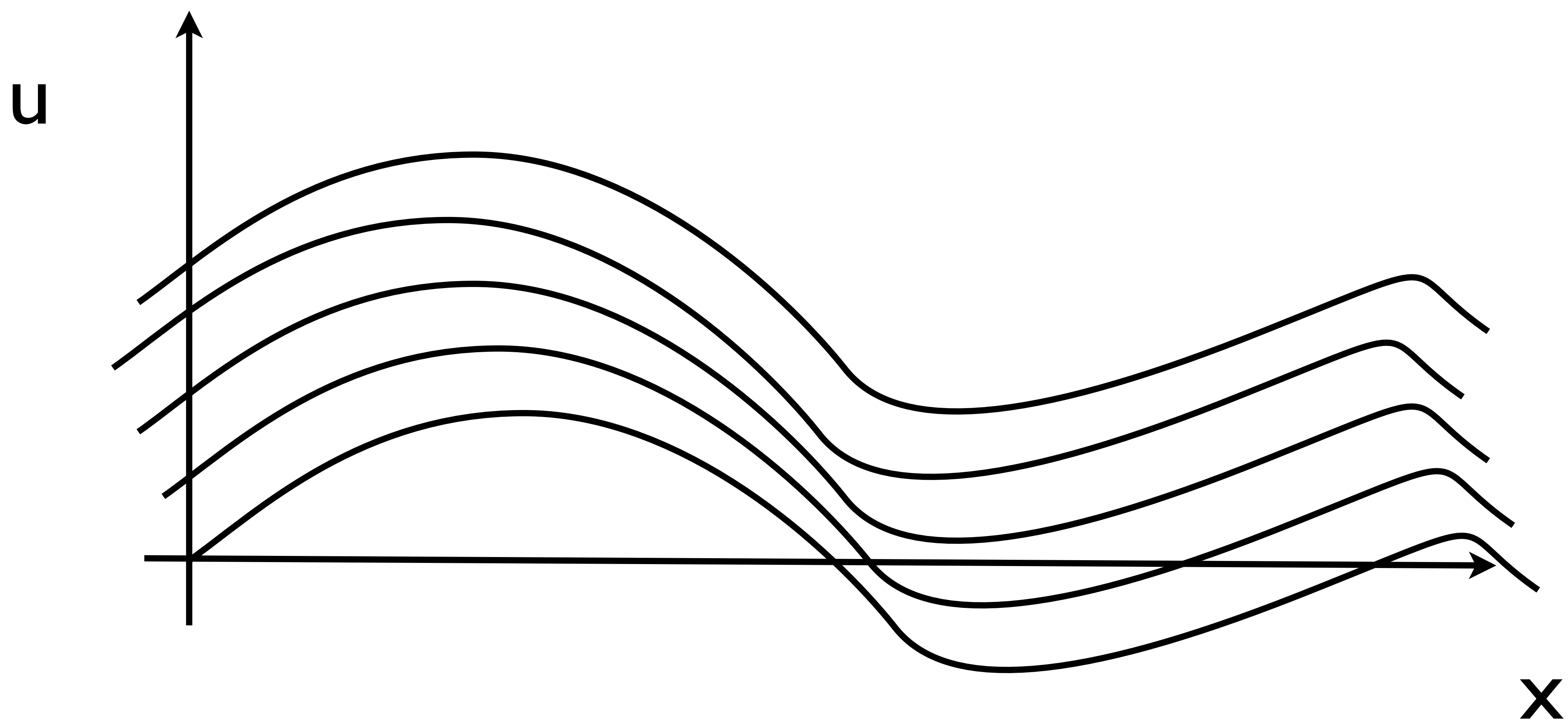
function $u = f(x,t)$ →  → reward $C(x(t), u(t))$



system dynamics
$$\dot{x} = f(x,t)$$

Dynamic optimization

'optimization of functions'



Optimal control

in continuous stochastic state-action spaces

(gaussian)
noise

$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t, t) + \mathbf{G}(\mathbf{x}_t)\,(\mathbf{u}_t + \epsilon_t)$$

state vector

controls

input gain matrix

system dynamics (nonlinear)

A D R L

ETH Zürich

Optimal control
in continuous stochastic state-action spaces

Find u that maximizes reward (minimizes cost)

$$R(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t \, dt$$

given diff. constraints (system dynamics)

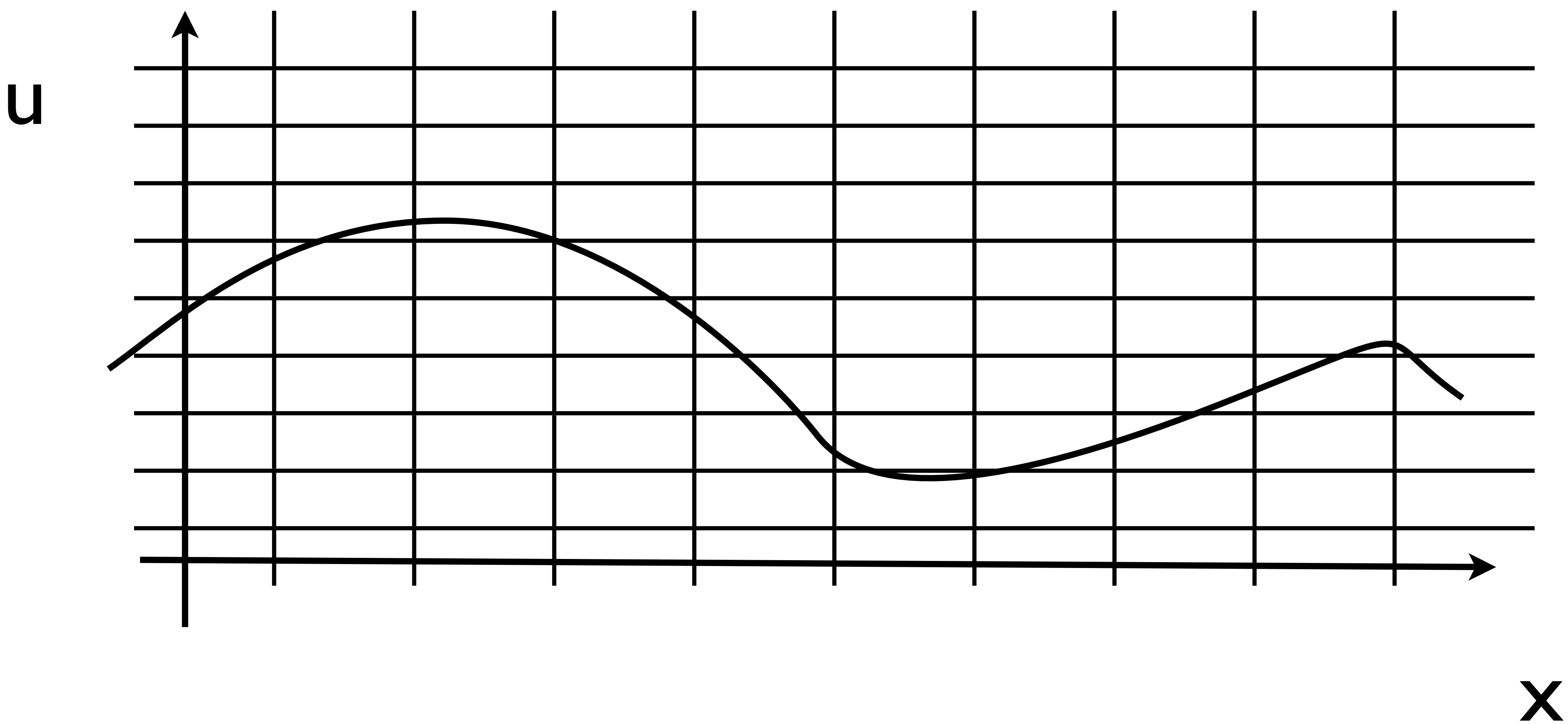
$$\dot{\mathbf{x}}_t = \mathbf{f}(\mathbf{x}_t, t) + \mathbf{G}(\mathbf{x}_t)\left(\mathbf{u}_t + \epsilon_t\right)$$

A D R L

ETH Zürich

Control Policies

Naive: state - action mapping



Problem: dimensions!

Cost and Reward

'how bad is a solution' - cost
'how good is a solution' - reward

$$C = -R$$

all is relative: costs/rewards can have arbitrary offsets
learning progress is measured relative to previous costs/rewards

Cost and Reward

(Accumulated/Total) Cost

$$R(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} r_t \, dt$$

Final cost

Intermediate cost

$$r_t = r(\mathbf{x}_t, \mathbf{u}_t, t) = q_t + \frac{1}{2}\mathbf{u}_t^T \mathbf{R} \mathbf{u}_t$$

Interm. cost examples

$$r_{t=1.5s}^{waypoint} = |\mathbf{x}_{t=1.5s} - \mathbf{x}^{waypoint}|$$

$$r_t^{falling} = 1$$

$$r_t^{CoP} = \frac{1}{N}|\mathbf{c}_t - \mathbf{c}^{default}|$$

Final cost

$$\phi_{t_N} = 10^4 \cdot (\psi_{max} - \psi_N)$$

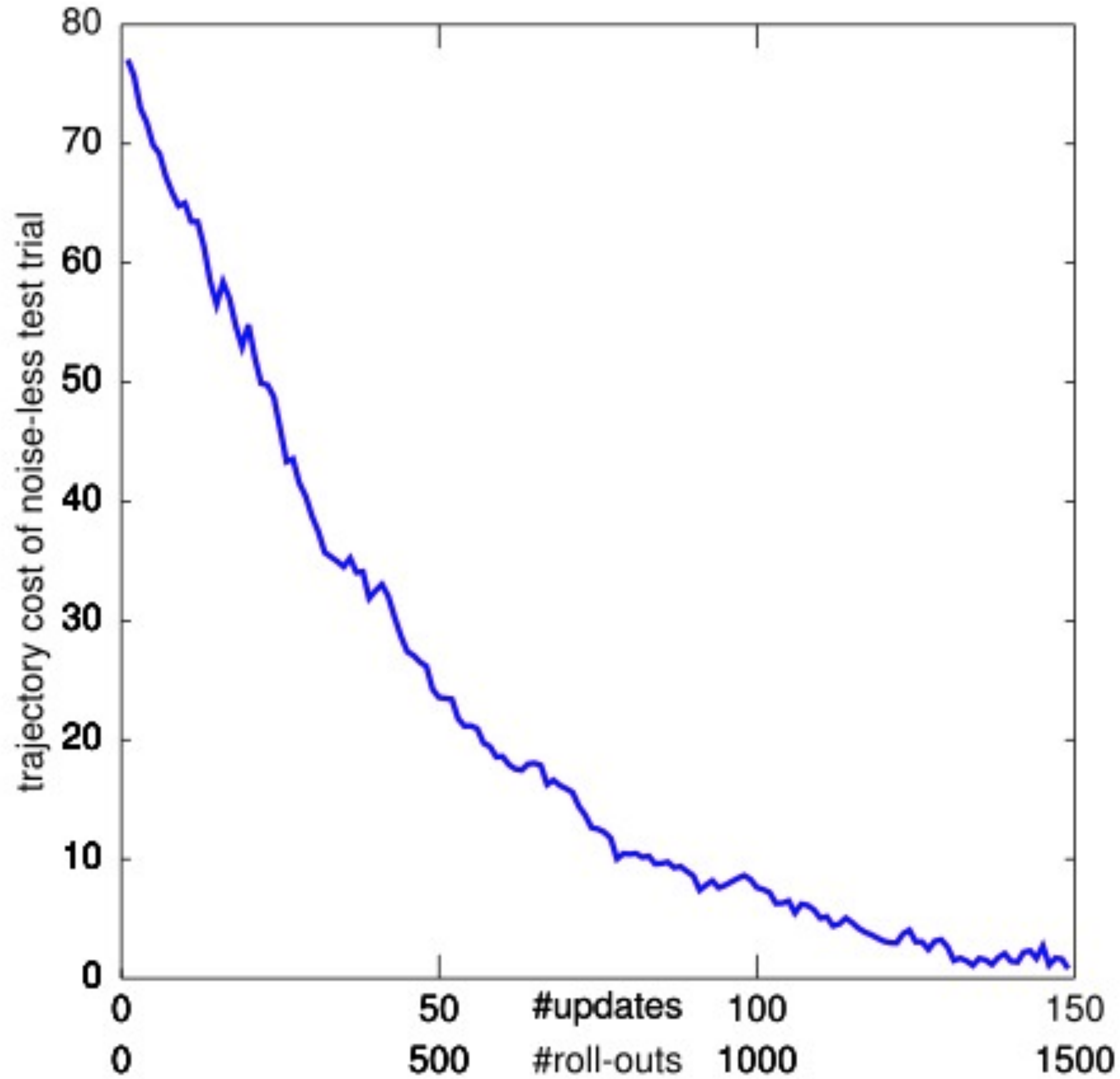
A D R L

$$\mathbf{u} = -\mathbf{K}_P(\mathbf{q} - \mathbf{q}_d) - \mathbf{K}_D(\dot{\mathbf{q}} - \dot{\mathbf{q}}_d) + \mathbf{u}_{ff}$$

ETH Zürich

Learning curves

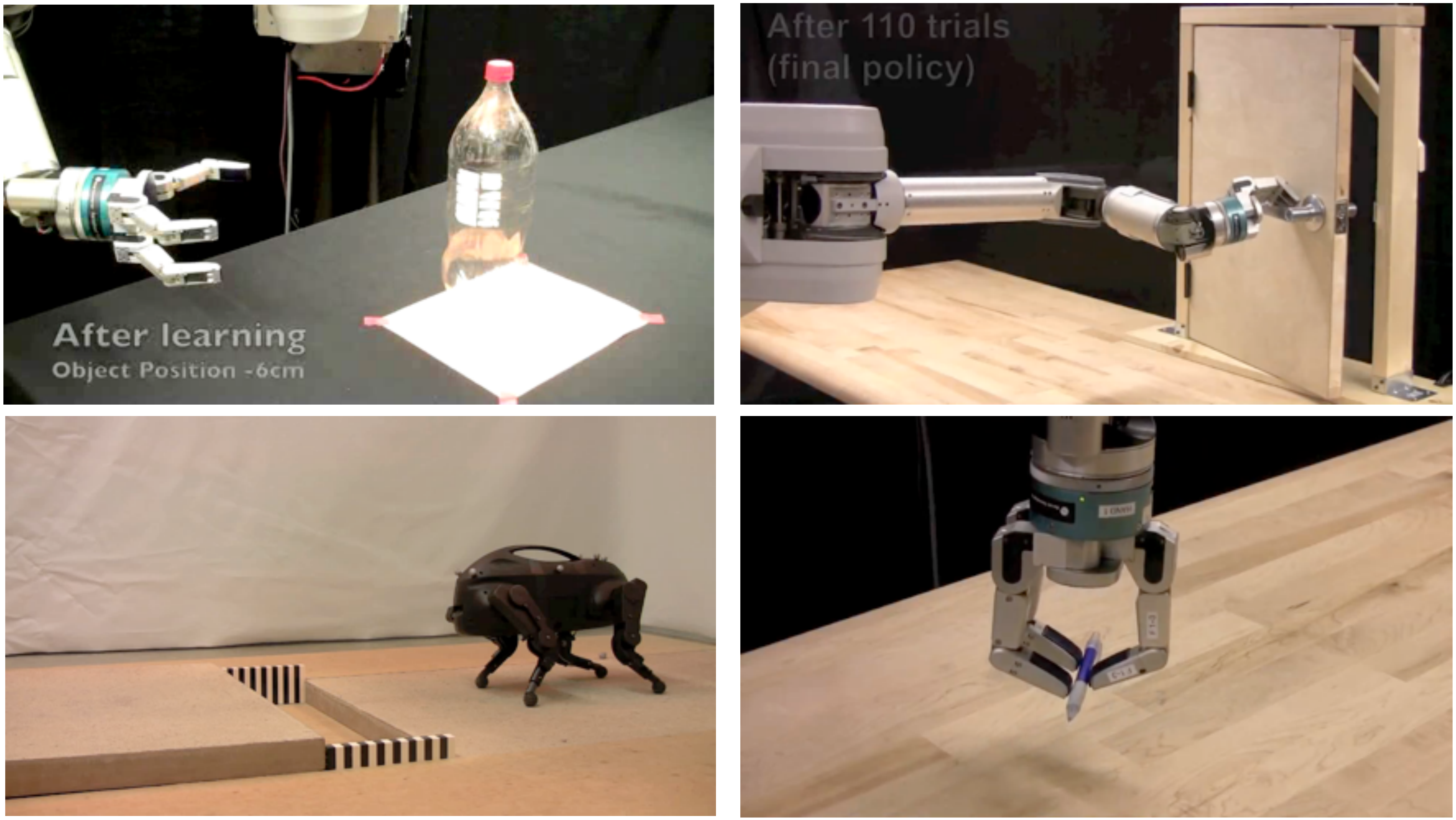
Reward
Cost



'trials'

Learning Complex Movement Skills

CLMC Lab



A D R L

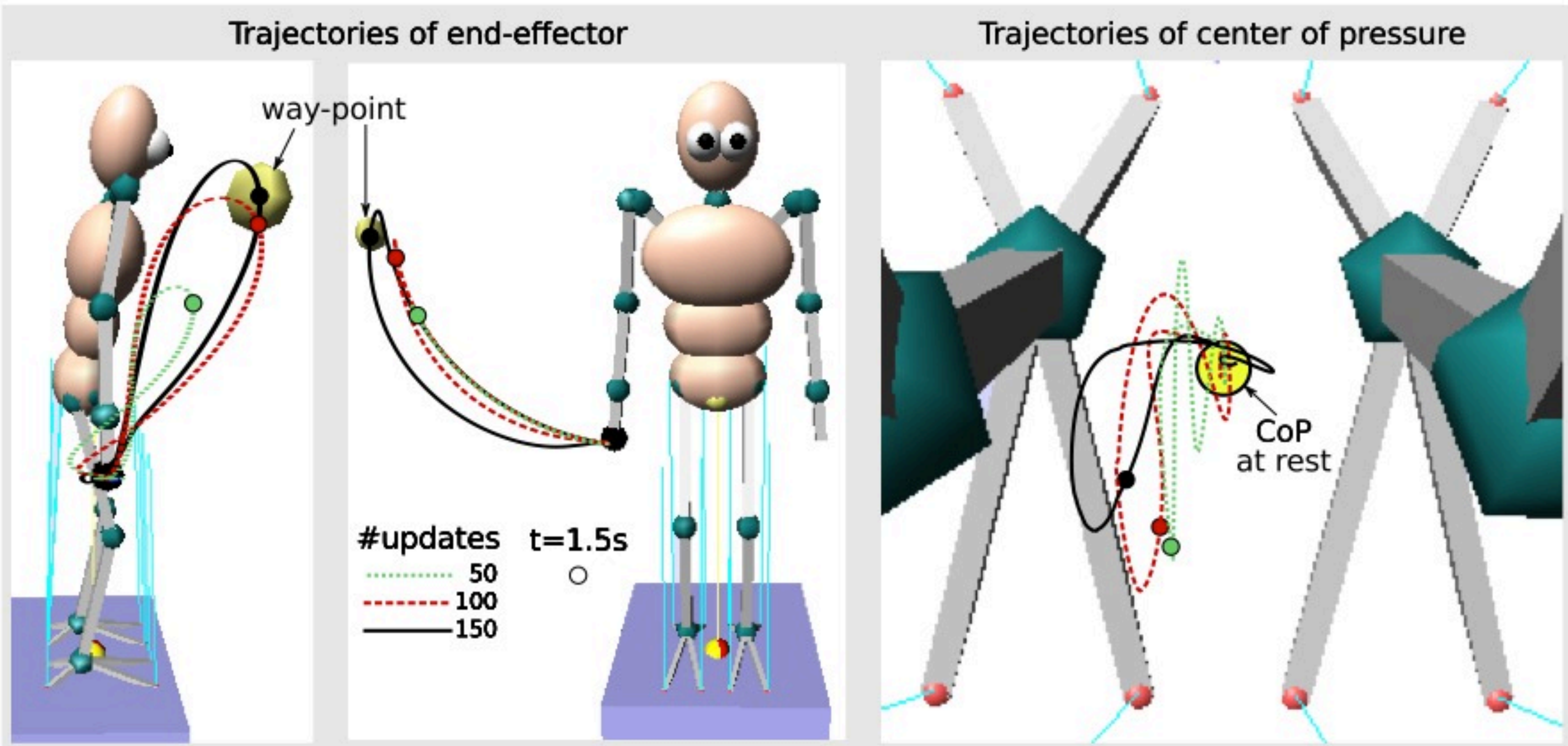
ETH Zürich

High-dimensional tasks

'whole body' skills - high dimensions
conflicting tasks (e.g. balance and reaching)

Reinforcement Learning of Full-body Humanoid Motor Skills

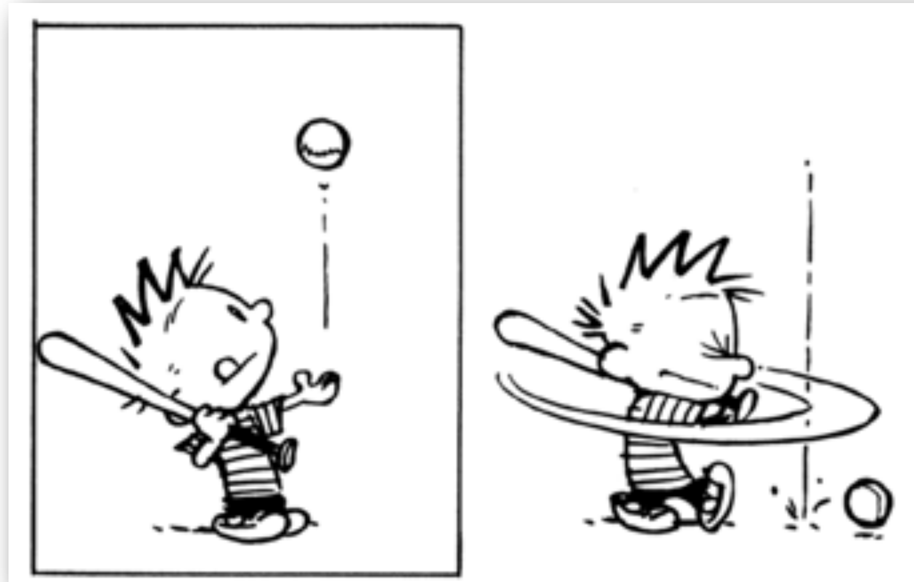
Freek Stulp, Jonas Buchli, Evangelos Theodorou, Stefan Schaal



$$r_t^{CoP} = \frac{1}{N}|\mathbf{c}_t - \mathbf{c}^{default}|$$

ADRL

ETH Zürich



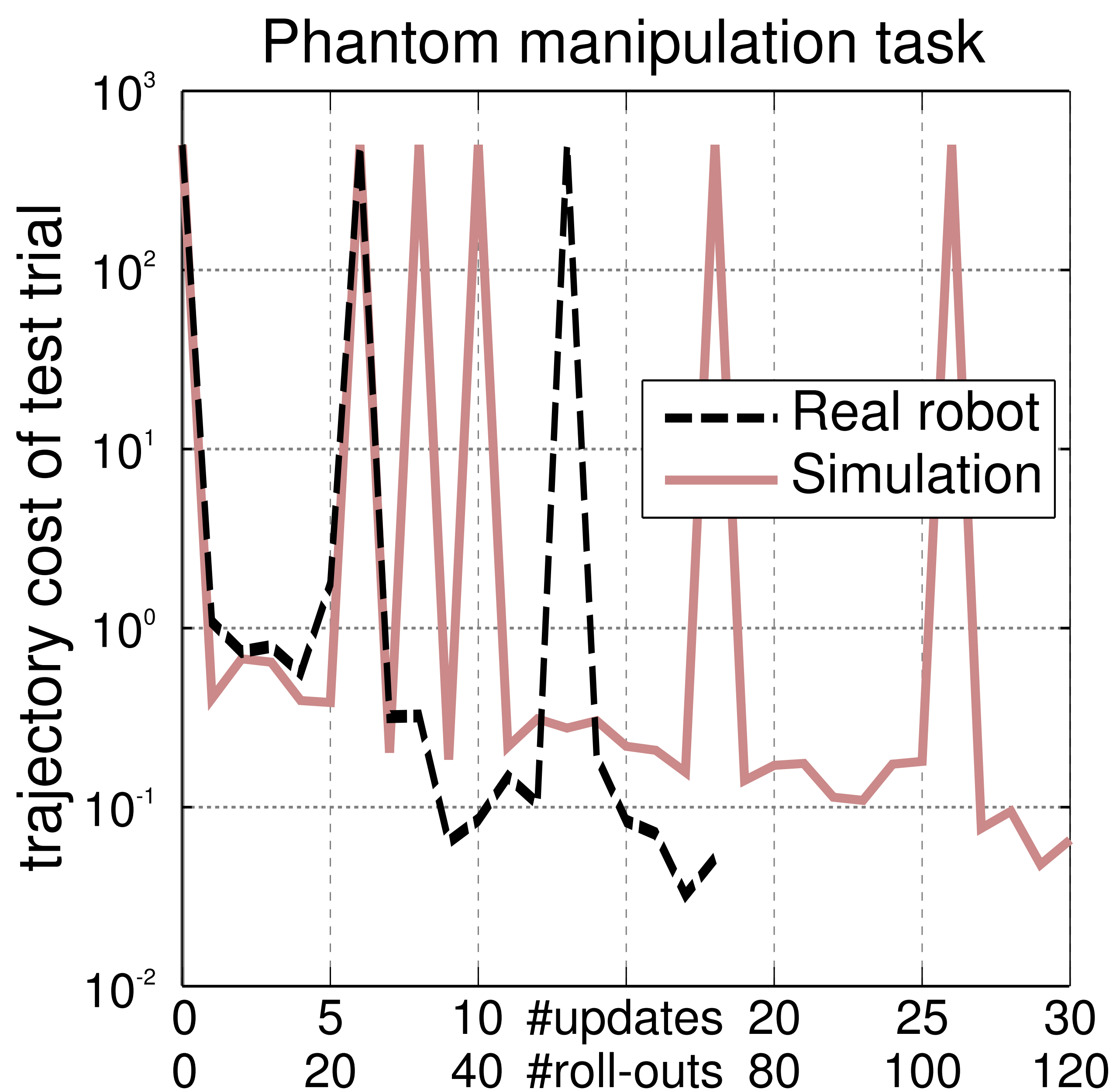
RL: unspecific feedback!

Discrete reward

Discrete reward

Learning Variable Impedance Control

Jonas Buchli*, Freek Stulp, Evangelos Theodorou, Stefan Schaal †



Goal: Switch on light and use lowest amount of effort

$$r_t = \frac{1}{N}\sum_{i=1}^{3} K_{P,t}^{i}$$

The terminal cost ϕ_{t_N} is 0 if the switch was flipped, or 500 if it was not.

Learning of force policies

Learning Force Control Policies for Compliant Manipulation

Mrinal Kalakrishnan*, Ludovic Righetti*, Peter Pastor*, and Stefan Schaal*†

Learning Force Control Policies for Compliant Manipulation

Mrinal Kalakrishnan*, Ludovic Righetti*, Peter Pastor*, and Stefan Schaal*†

Learning Force Control Policies for Compliant Manipulation

Mrinal Kalakrishnan, Ludovic Righetti, Peter Pastor, Stefan Schaal

CLMC Lab, University of Southern California

www-clmc.usc.edu

Cost functions

Door:

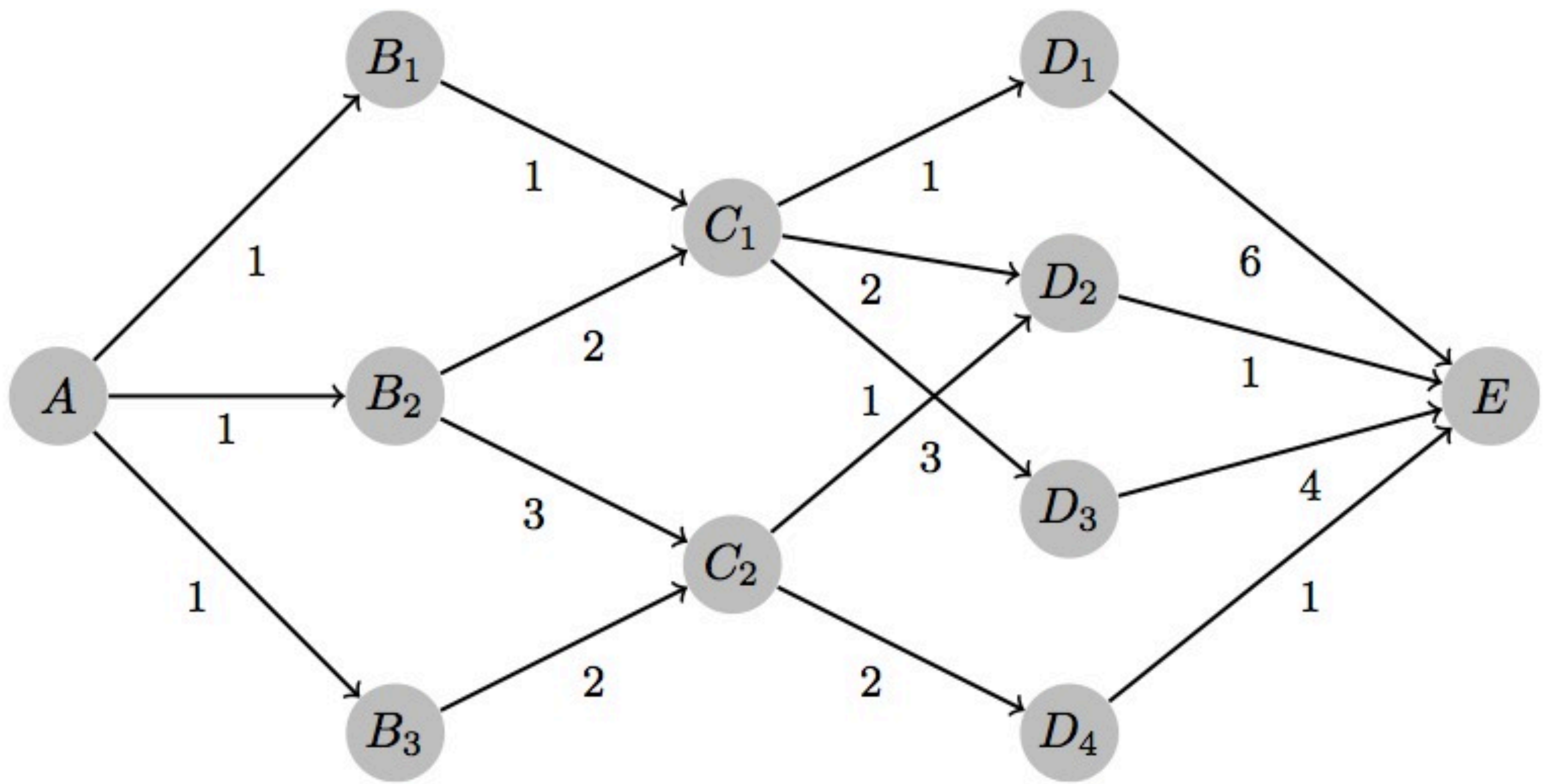
cost function at time t is: $r_t = 300q_{door} + 100q_{handle} + 100q_{pos} + 10q_{orient} + 0.1q_{fmag} + 0.02q_{tmag} + 0.02q_{ttrack} + 0.01q_{ftrack} + 0.0001\boldsymbol{\theta}^{\mathrm{T}}\mathbf{R}\boldsymbol{\theta}$, where q_{door} and q_{handle} are the squared tracking errors of the door and handle angles respectively, q_{pos} and q_{orient} are the squared tracking errors of the position and orientation of the hand, q_{fmag} and q_{tmag} are the squared magnitudes of the desired forces and torques, q_{ftrack} and q_{ttrack} are the squared force and torque tracking errors, and $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{R}\boldsymbol{\theta}$ is the control cost.

Pen:

into 100 time-steps. The immediate cost function at time t is: $r_t = 100q_{pen} + 1.0q_{ftrack} + 0.5q_{fingertrack} + 0.1q_{fmag} + 0.0001\boldsymbol{\theta}^{\mathrm{T}}\mathbf{R}\boldsymbol{\theta}$, where q_{pen} is an indicator cost which is 1 if the pen has slipped out of the hand (as described above), q_{ftrack} is the squared force tracking error, $q_{fingertrack}$ is the squared finger position tracking error, q_{fmag} is the squared force magnitude, and $\boldsymbol{\theta}^{\mathrm{T}}\mathbf{R}\boldsymbol{\theta}$ is the control cost. After 90 trials, we

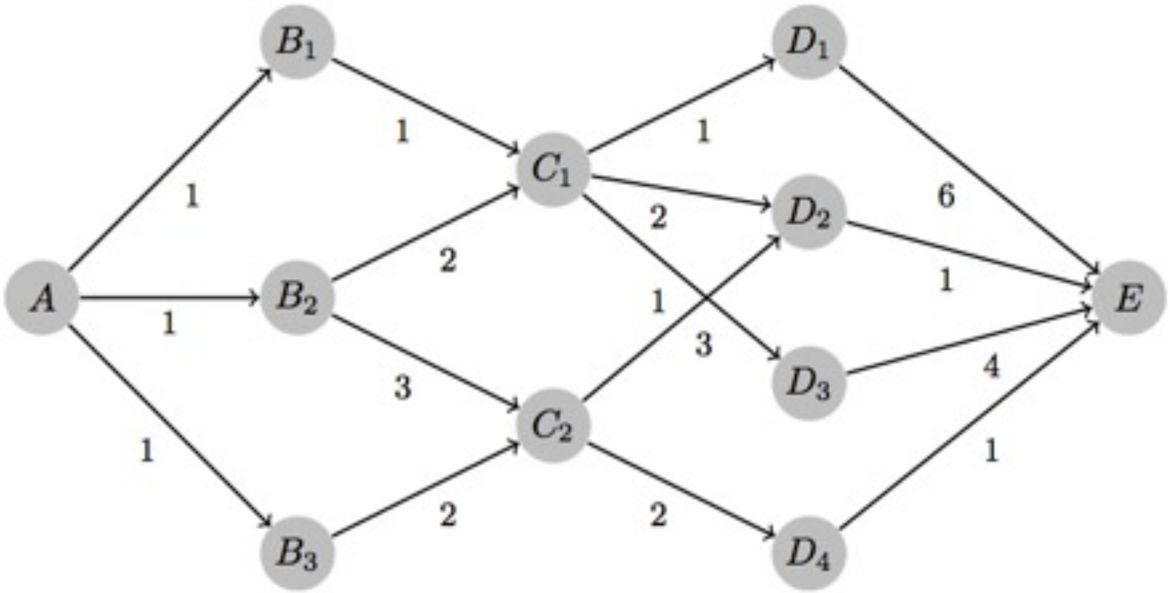
Principle of optimality

Traveling Salesman



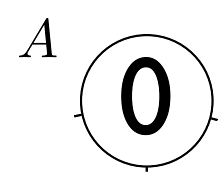
11 nodes

16 edges



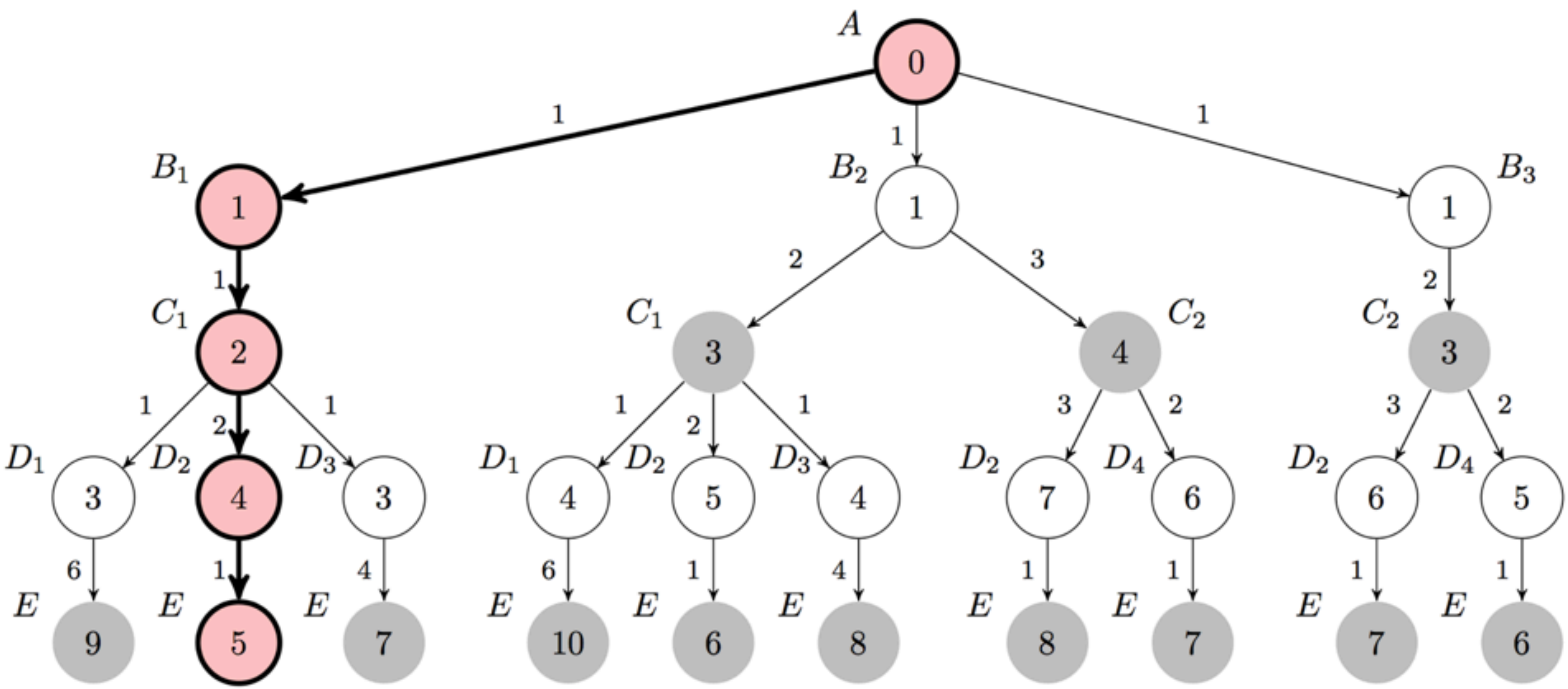
$$V(n) = \sum_{i=0}^{n} e(i)$$

Accumulated path cost at each node



forward decision tree

Forward tree



10 paths

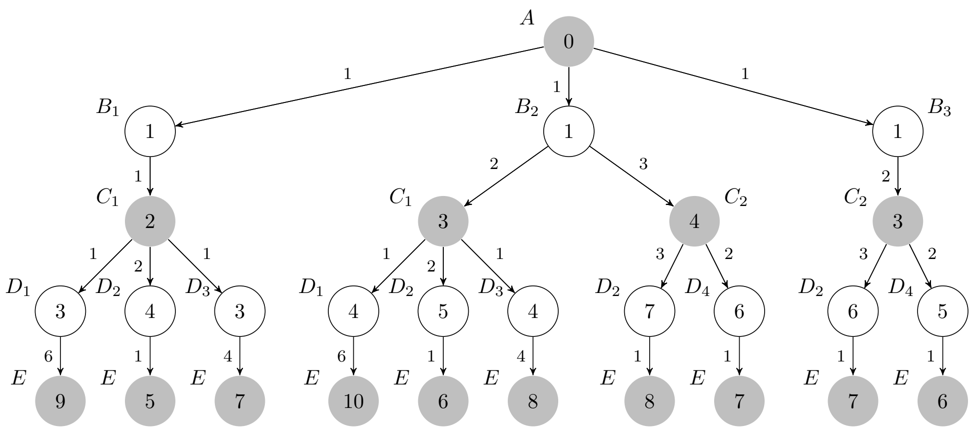
28 nodes
27 edges

A D R L

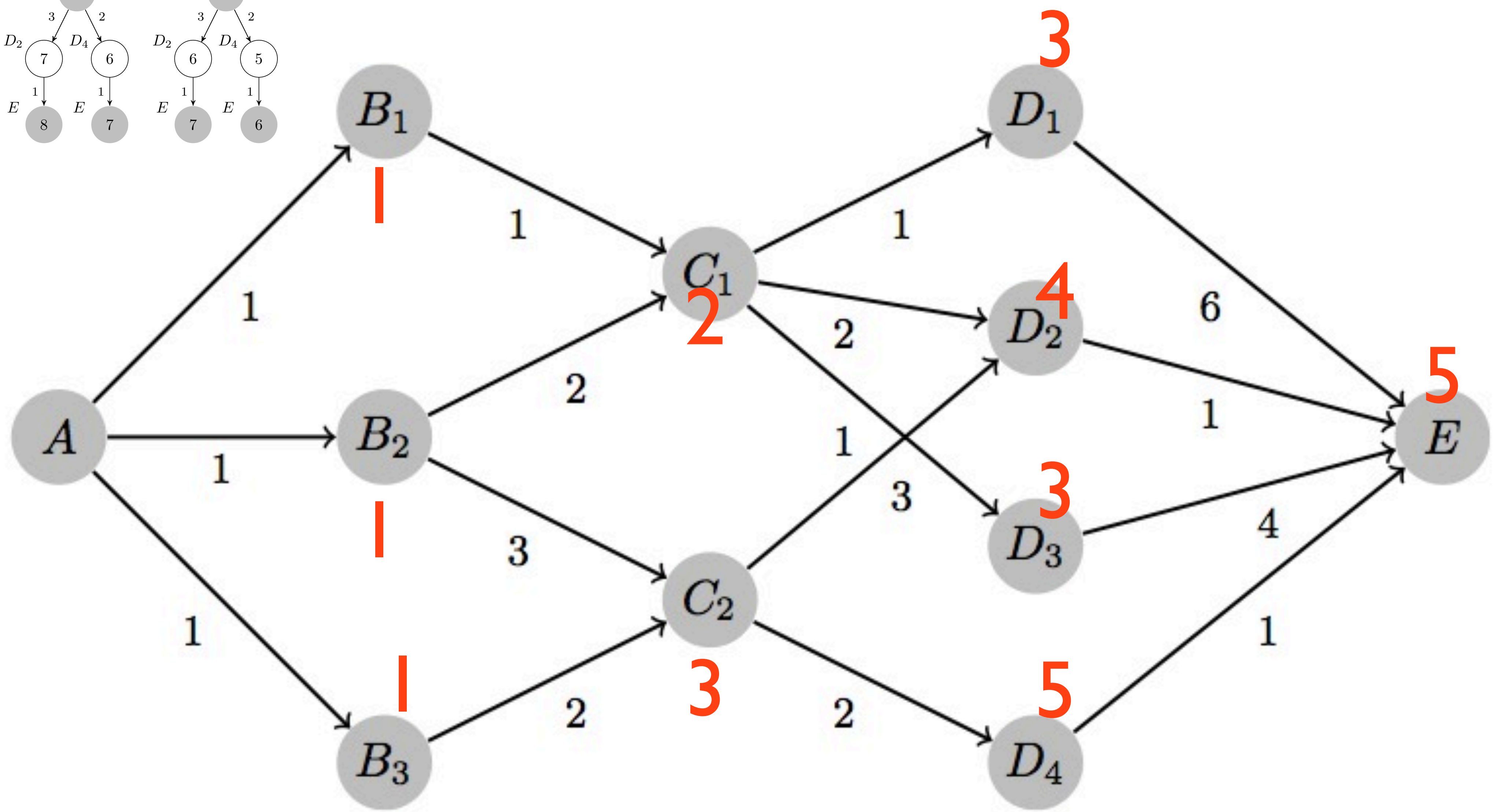
ETH Zürich

= ‘decision making’/‘control’

Select optimal path?



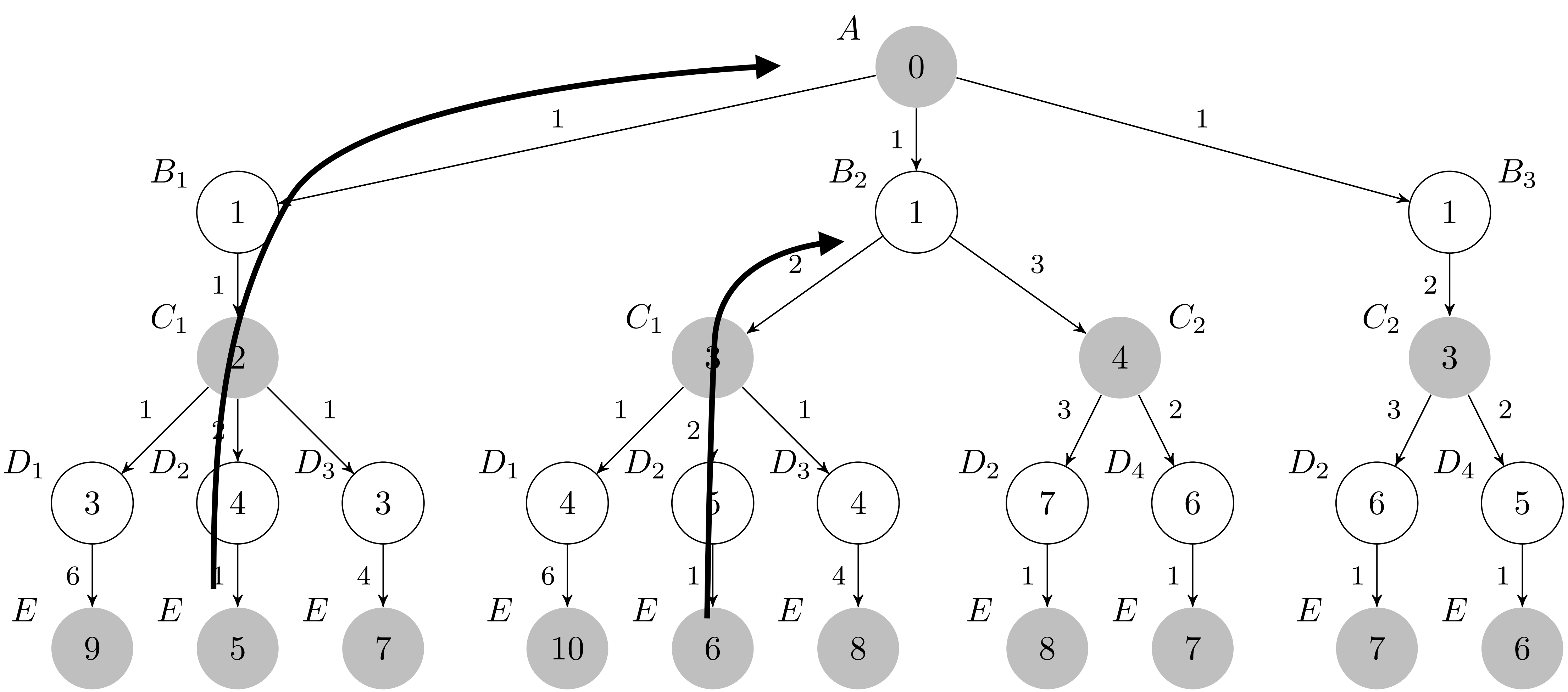
‘map with shortest possible distance driven thus far’



need to look all the way to the end to find optimal path, local info (edge or next node is not informative)

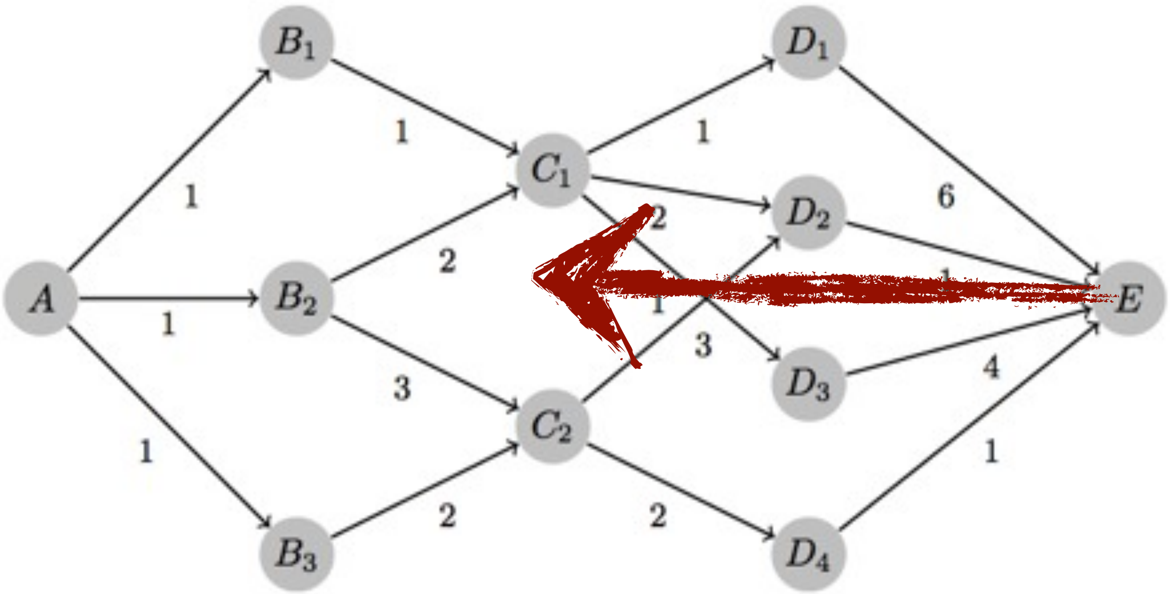
Select optimal path?

= control



need to look all the way to the end to find optimal path, local info (edge or next node is not telling)

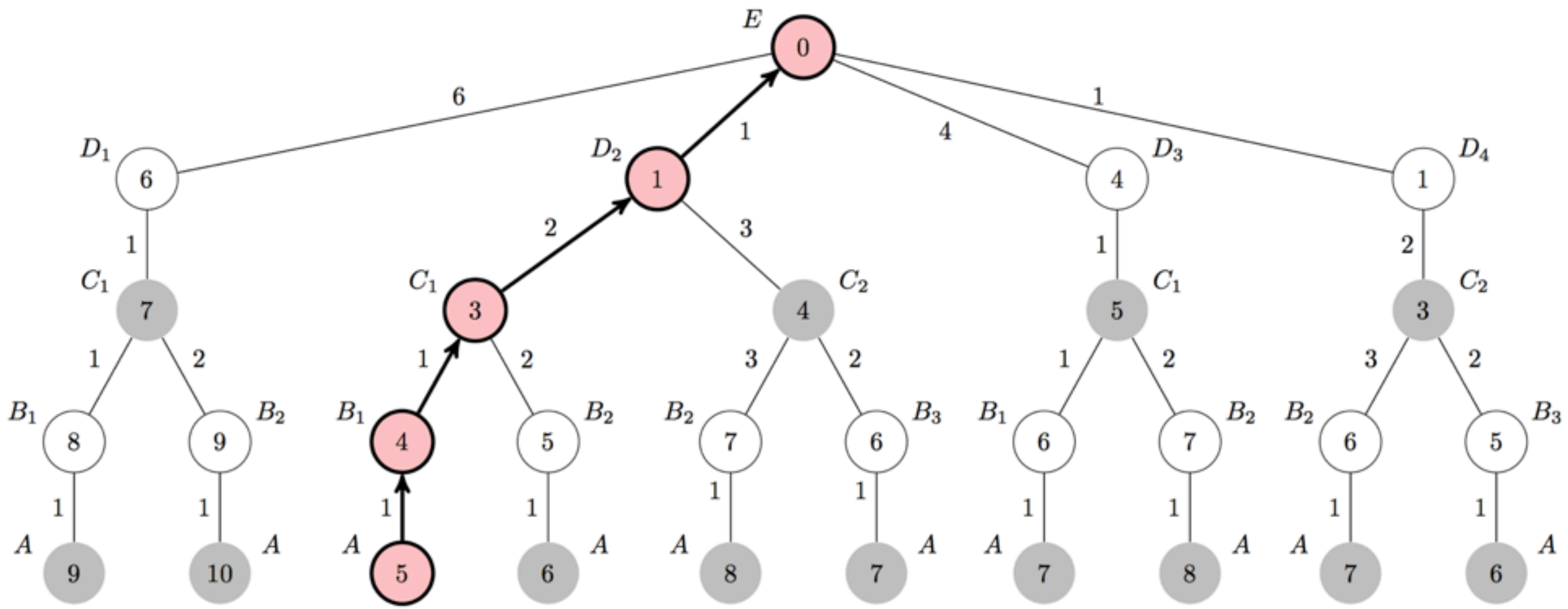
Backward tree



$$V(n) = \sum_{i=n}^{N} e(i)$$

$$V(n) = \sum_{i=N}^{n} e(i)$$

Backward tree



10 paths

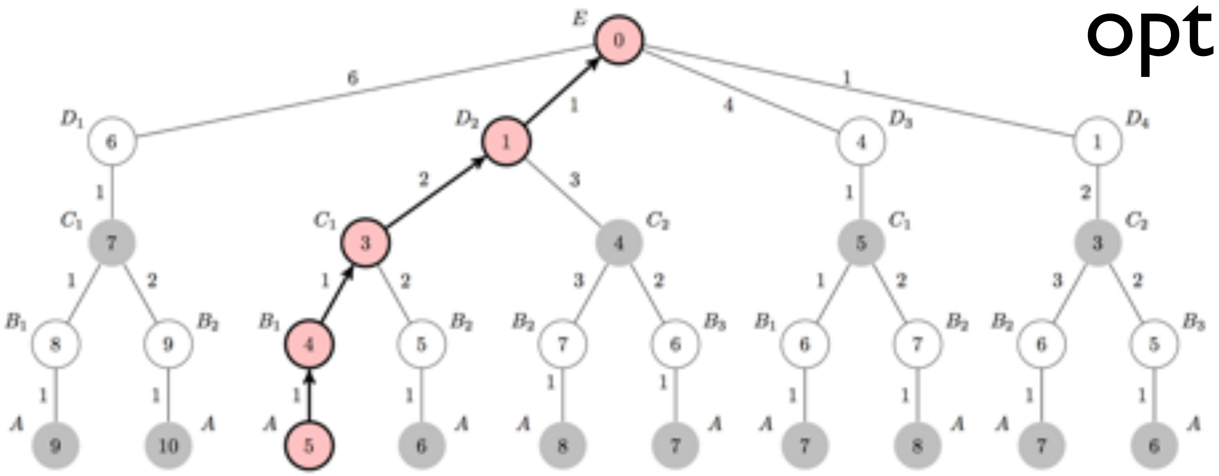
30 nodes
29 edges

ADRL

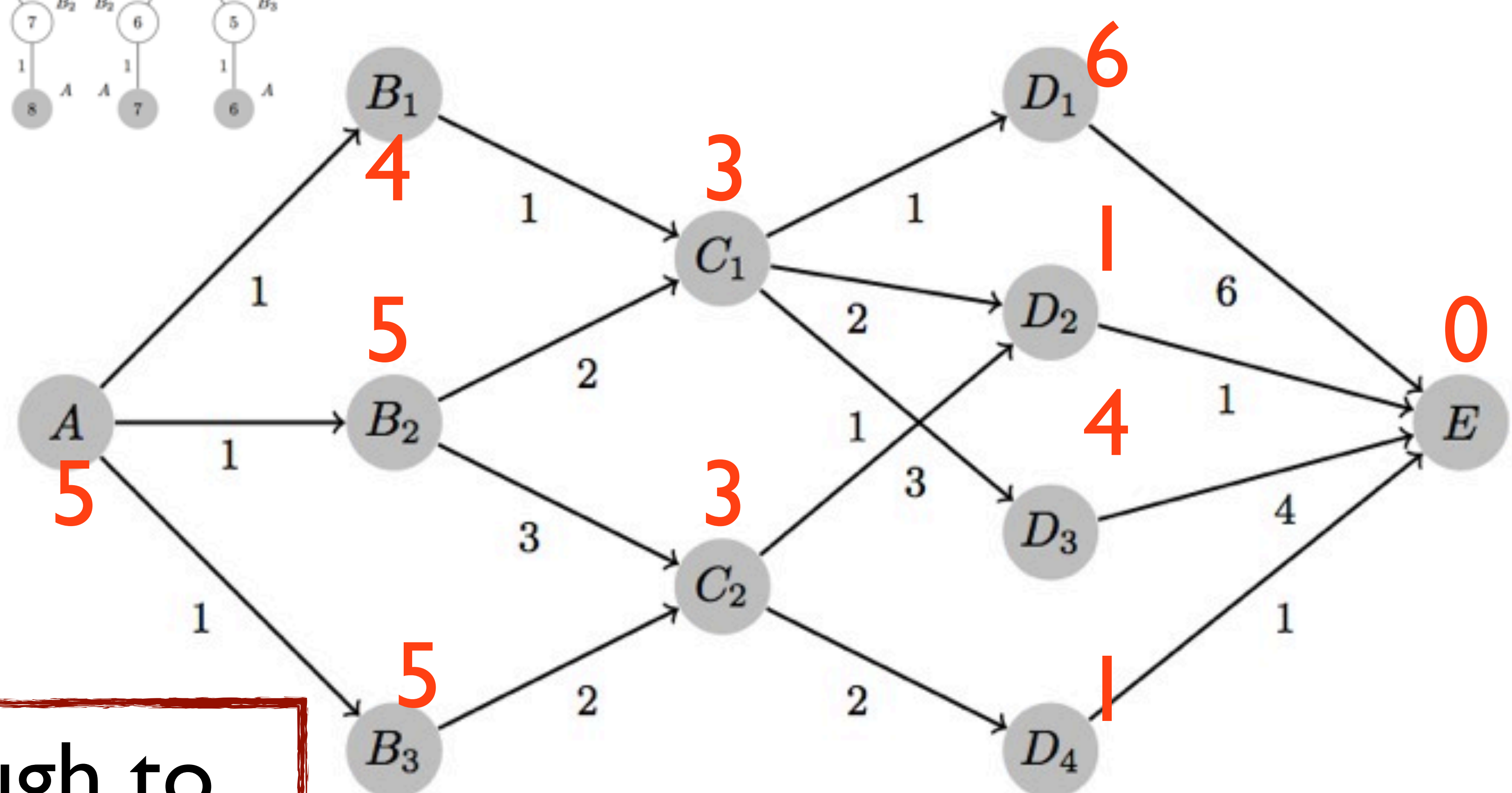
ETH Zürich

Map of shortest achievable distance

'How long is the shortest path from here to goal if following the optimal route?'



'How valuable is a position?'



'local info' is enough to determine next step!!!

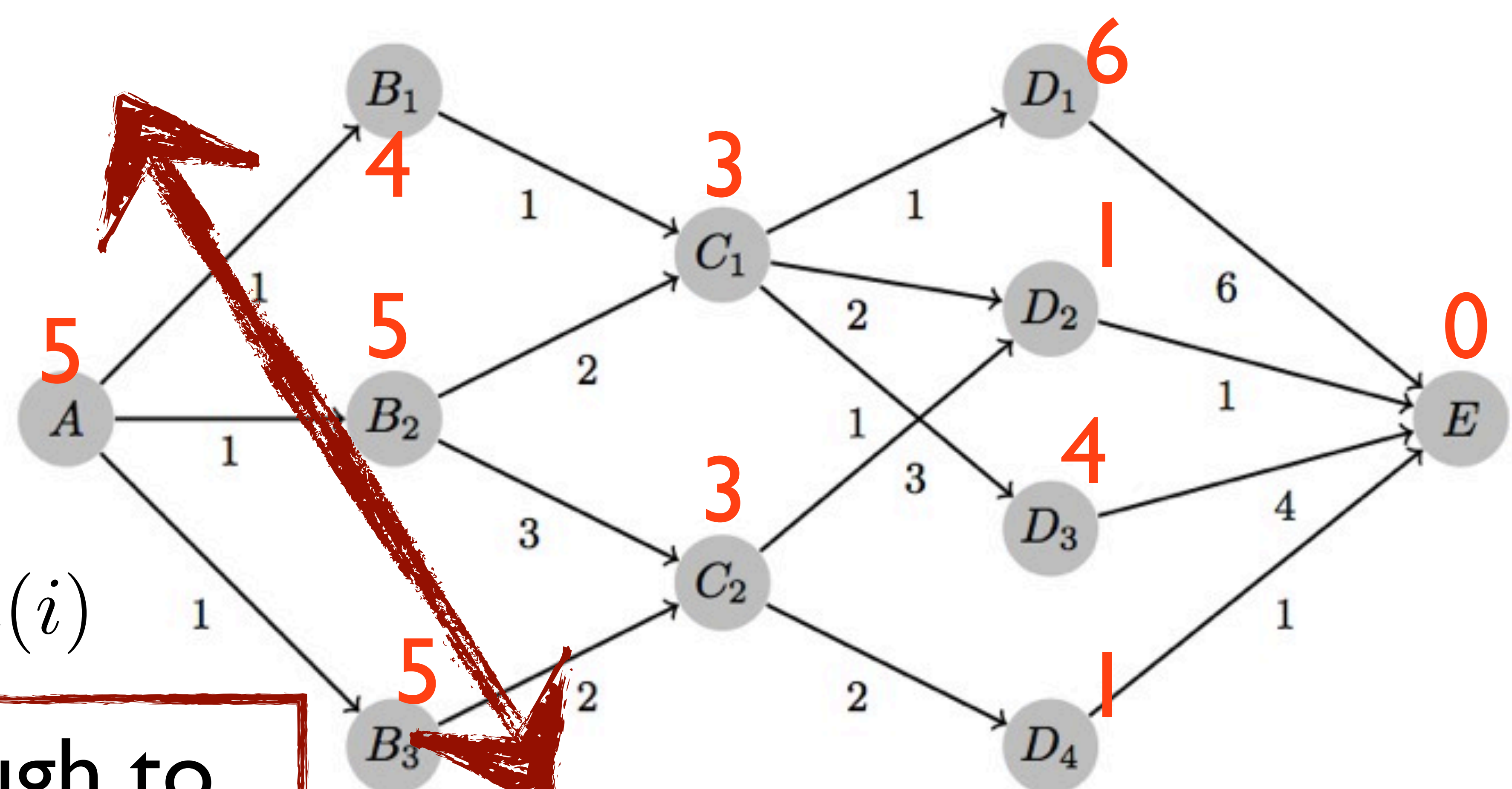
Map of shortest achievable distance

'How long is the shortest path from here if following the optimal route?'

'How valuable is a position?'
'What cost can I expect?'

$$V(n) = \sum_{i=N}^{n} e(i)$$

$$V(n) = e(n) + \sum_{i=N}^{n+1} e(i)$$



'local info' is enough to determine next step!!!

Value function

A D R L

ETH zürich

Principle of optimality

If a path ABCDE is optimal, then all parts of this path starting at intermediate position and ending at E (BCDE,CDE,DE) are optimal.

Finding optimal path/control as recursive problem or backwards search

THM

★Value function is very useful to find decision/control

★Value function is found by backward sweep (from goal to start)