Tutorial RL4AA'24

Concepts to overcome challenges in applying RL to accelerators
- from deep meta-RL to safe shallow model-based RL

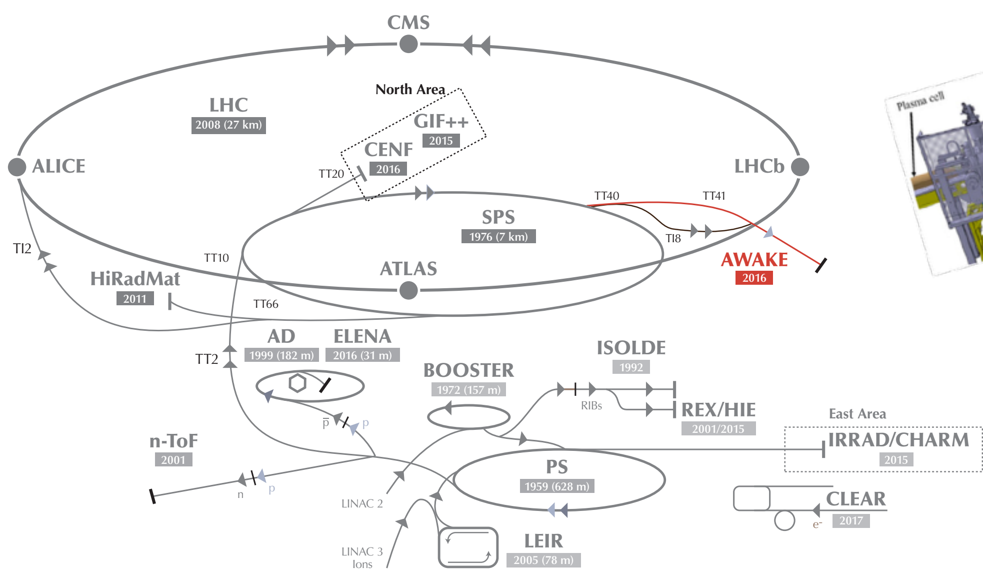
Simon Hirlaender, Sabrina Pochaba, Lukas Lamminger, Nico Madysa, Andrea Santamaria Garcia, Jan Kaiser, Chenran Xu, Annika Eichler

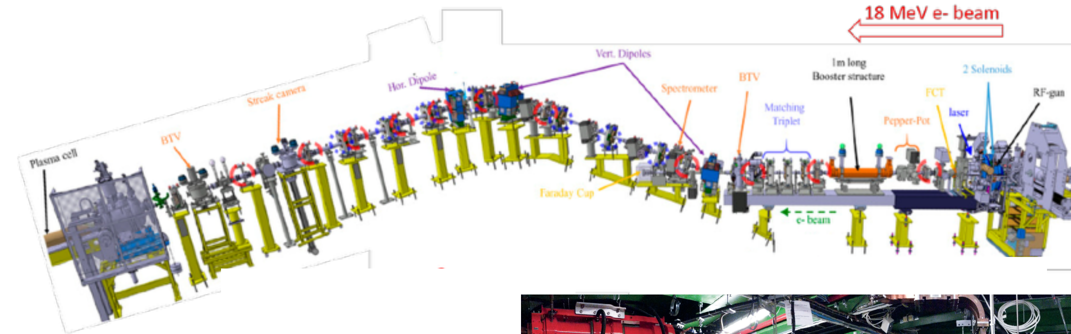
Goal of this tutorial

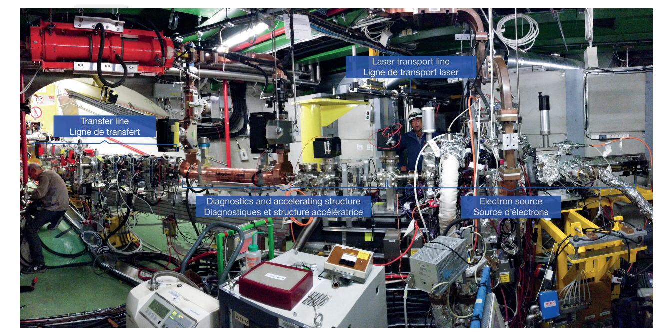
- Extend our toolbox - there is no one-fits all solution
- Give you concepts at the boundary of RL
- Fresh ideas to attack your RL problem you should be aware of

Problem set up

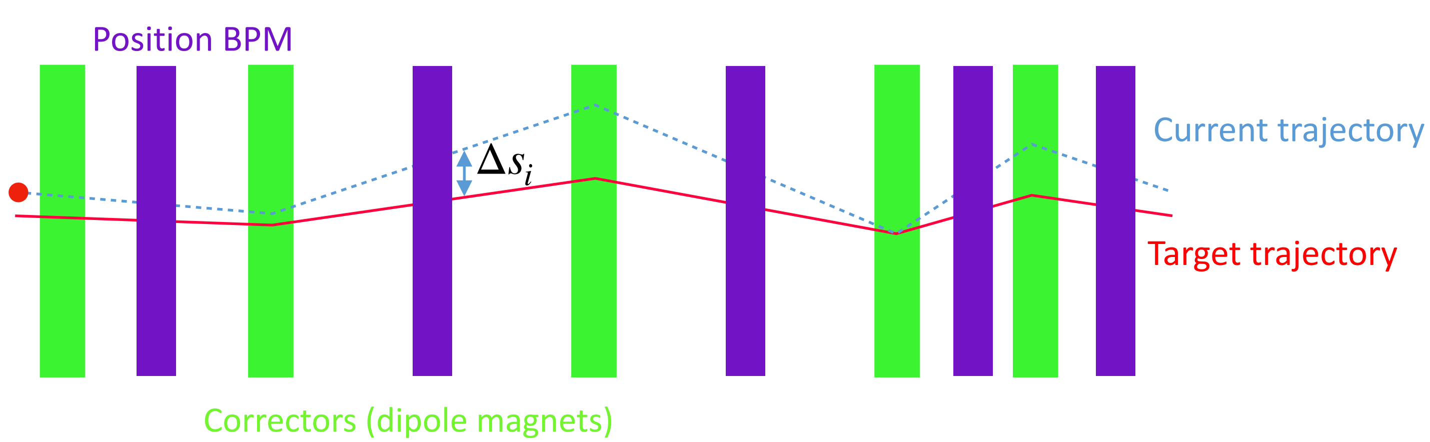
CERN AWAKE steering problem







- AWAKE electrons - start 5 MV (RF gun), accelerated to 18 MeV transported through beam line of 12 m to the AWAKE plasma cell.
- Vertical 1 m step and a 60° bend bring electron beam parallel SPS proton beam shortly plasma cell.
- The trajectory is controlled with 10 horizontal and 10 vertical steering dipoles according to the measurements of 10 beam position monitors (BPMs).

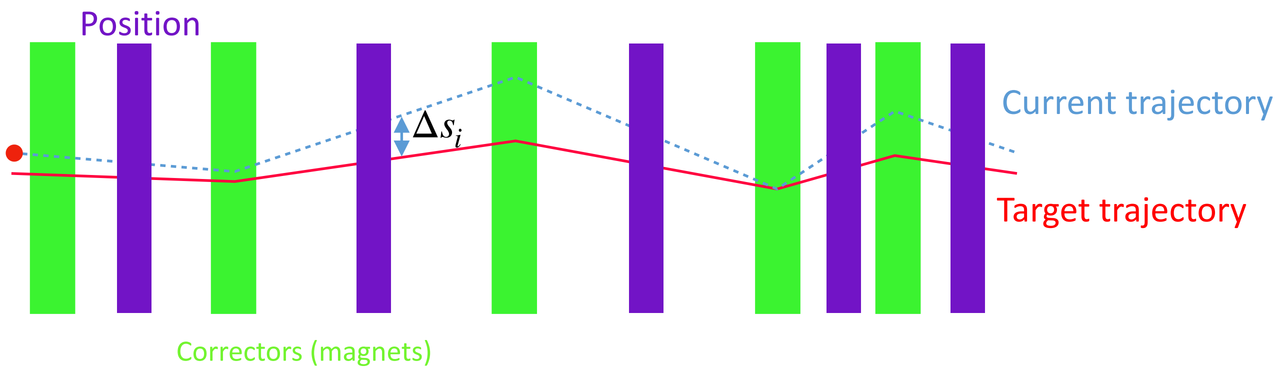


CERN AWAKE steering problem

- Well studied in several papers/thesis
- Linear Dynamics with 10 degrees of freedom
- Non-trivial due to action limitations
- Analytical solution for the optimal policy
- Easy to understand, focus on the RL problem not the MDP
- The <u>simulation corresponds exactly to the real system</u> (measured optics)
- <u>All our algorithms were tested on the real machine</u>

CERN AWAKE steering problem MDP

Markov decision process: $(S, A, R, P, \rho_0, \gamma)$

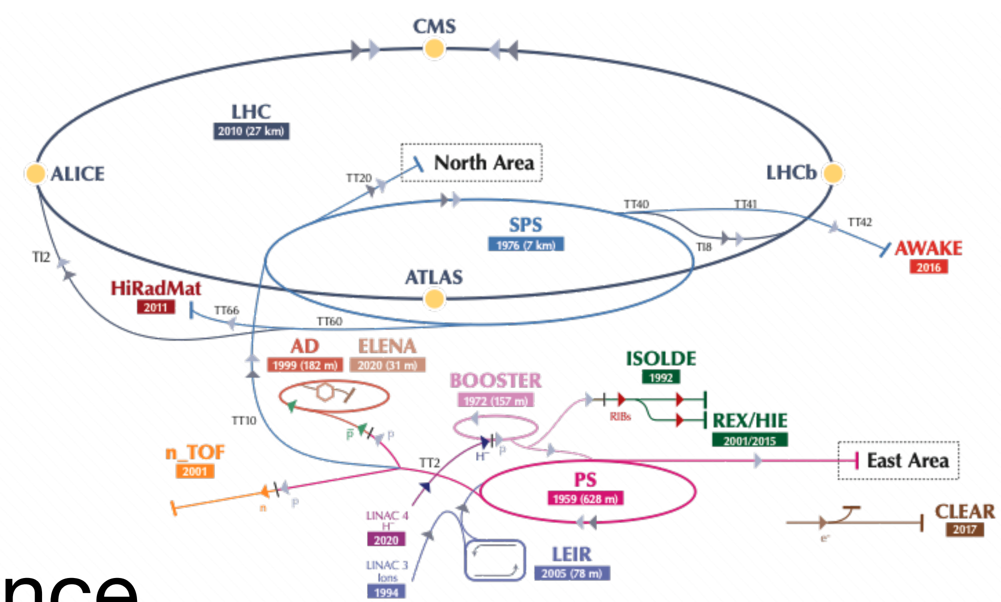


- 10 continuous states S and actions $A \in [-1,1]$ (10 DoF problem - observation is state)
- Rewards R negative of RMS of states $r_i \propto -\sqrt{\sum \Delta s_i^2}$
- Actions are done in $s_{t+1} = \mathbf{R}a_t + s_t$
- Episodic training
- Initial criteria: Initial distribution ρ_0 is away from low RMS - to make problem a bit challenging
- Termination criteria:
  - Maximal number of interactions (truncation)
  - RMS below measurement uncertainty
  - States s_i > beam pipe
- Transitions P are deterministic, γ =1
- If we speak about different tasks i (MPDs) we mean different optics $\mathbf{R}_i$

Motivation

RL in accelerator control

- Goals:
  - ➡ Set performance
  - ➡ Quickly recover performance
  - ➡ Maintain performance
  - ➡ Adapt to user changes



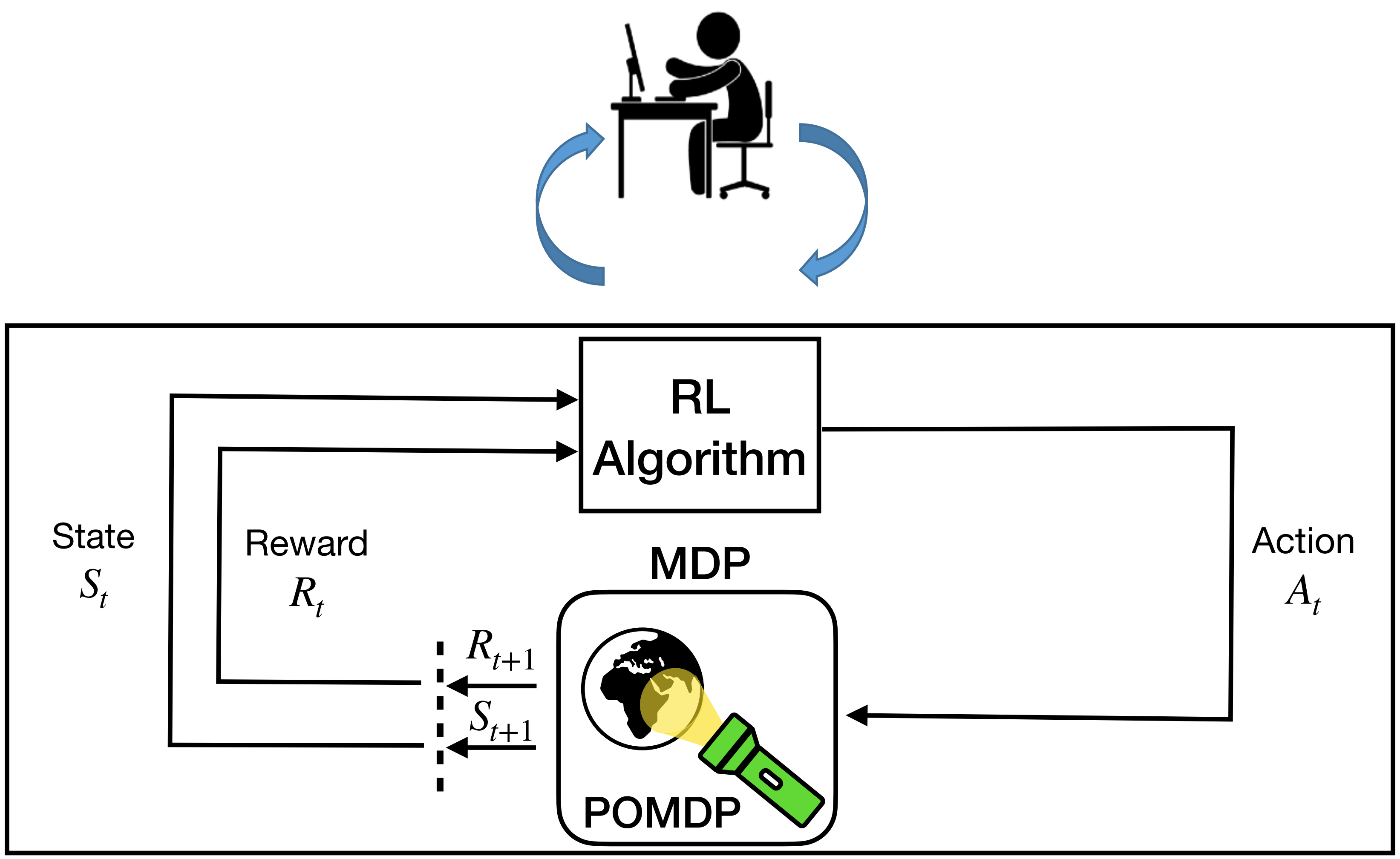
RL and accelerators - still rare

Effectively understand and optimise require significant expertise and computational resources.

Challenges and problems, both the RL algorithms and the physical system

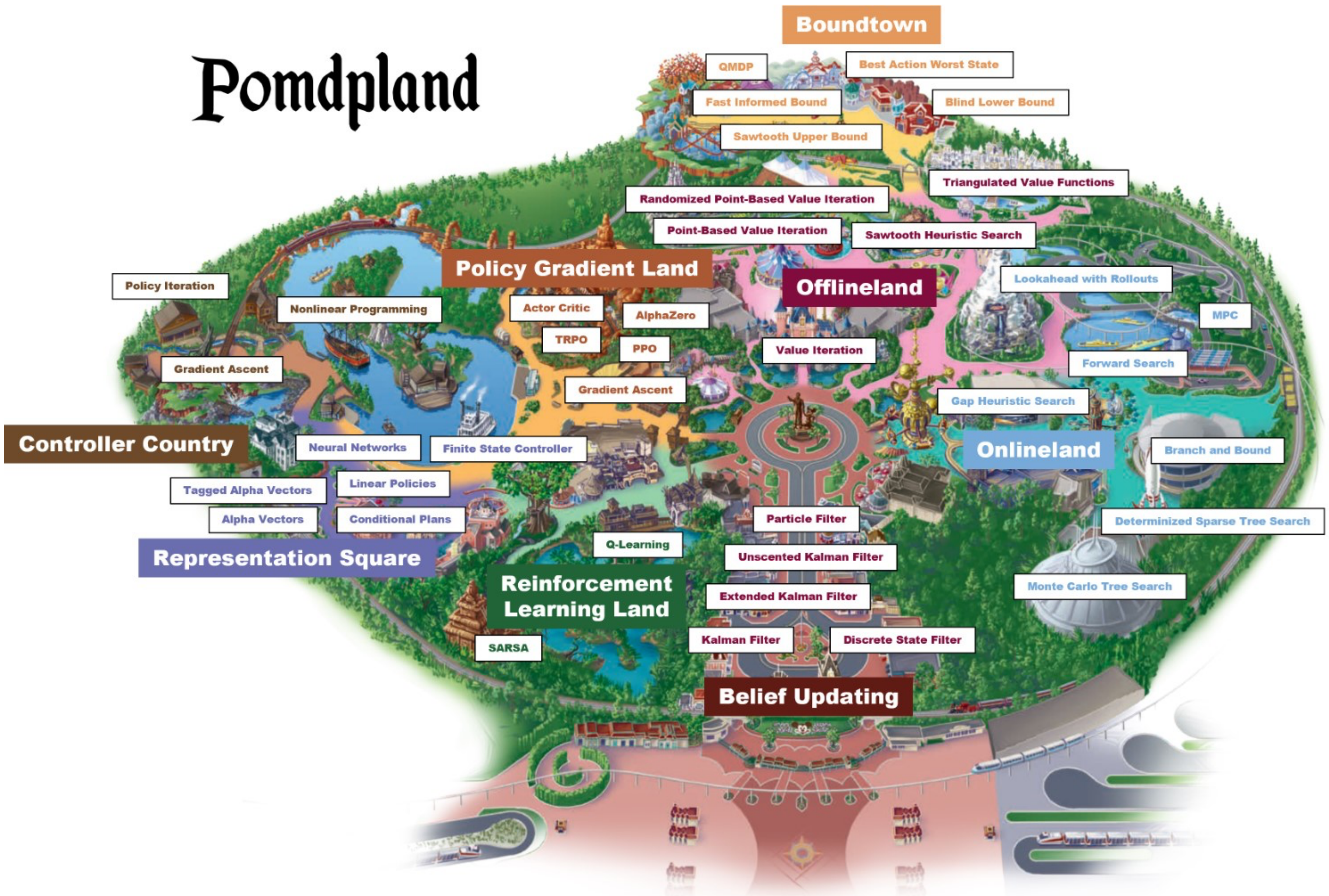
- Data Availability:
  - ➡ Slow and little data acquisition, maybe safety regulations
  - ➡ Modelling and Simulation Limitations
  - ➡ Long times needed to adjust after faults, resets, changes
- Integration with Existing Systems
- Long-term Stability and Maintenance
- General Safety and Reliability
- Real-Time Decision Making
- Computational Resources
- Generalisation and fast Adaptation

The entire problem



MDP Markov decision process
POMDP Partially observable Markov decision process

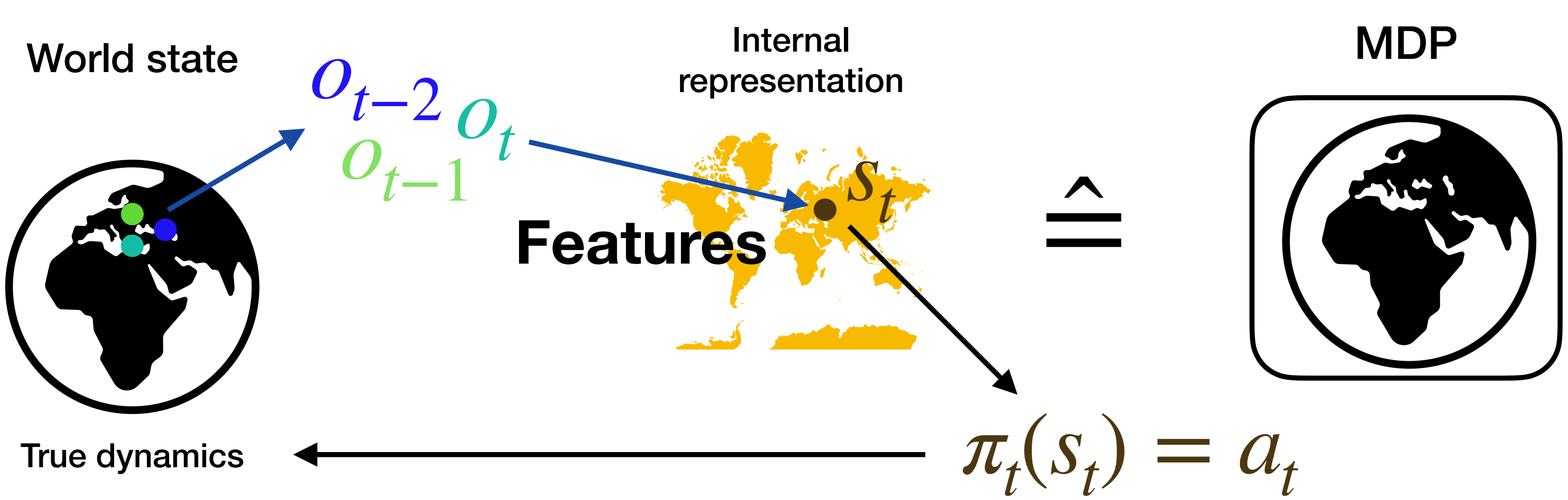
Wellcome to POMDPs



From Mykel Kochenderfer

Problem design - capture the right thing

- Solve an SDM problem: Information→Decision→Information→Decision→…
- Generally stochastic!
- Consequently we build a feedback system not planing too far in the future:
    - Define a **state** $s_t = h_t(o_t, a_{t-1}, o_{t-1}, a_{t-2}, o_{t-2} \ldots)$, as a function holding **sufficient statistics** until time step t for a decision - (example pong)
    - Decision based on s_t via: $a_t = \pi_t(s_t)$ - the policy - optimise an expected aggregate of future rewards



- Rarely the observation o is the state s, the world state is, but often we assume it is certainty equivalence!
- POMDP $\Rightarrow$ MDPs!

How bad is it?

- Linear POMDP: believe state - $O_t = h_t(S_t, A_t, W_t)$
  - ➡ Static output feedback is NP hard (linear in O_t and dynamics)
  - ➡ General POMDPs are PSPACE hard
- There are ways out - separation principle:
  - ➡ Filtering $\hat{s}_t = f(\{o_t\})$ - prediction problem
  - ➡ Action based on <u>certainty equivalence</u>
  - ➡ Optimal filtering - if dynamics are linear and noise is Gaussian - Kalman filtering - general belief propagation - LQG
  - ➡ Kalman filtered state - <u>optimal in estimation and control</u>
  - ➡ Estimate state with prediction $S_t = h(\tau_t)$, τ_t are time lags

POMDPs and non stationarity

- To find a proper state we have to solve the additional prediction problem $s_t = h_t(o_t, a_{t-1}, o_{t-1}, a_{t-2}, o_{t-2} \dots)$
- In the non-stationary, finite horizon formulation the MDP has the form $(S, A, \{P\}_h, \{r\}_h, H, \rho_0) \Rightarrow$ Value-functions $Q_h(s, a)$ get time depended $\Rightarrow$ similar form of Bellman equations
- We can incorporate time into state e.g. $\tilde{s} = (s, h) \Rightarrow$ standard MDP
- Generally Bellman equation nice in discounted, stationary formulation $\Rightarrow$ this is what we usually see and most libraries build on this formulation

Challenges of RL

1. Problem formulation - capturing the right problem in an MDP
   - ➡ State representation, Markov Property (e.g. non stationarity)
   - ➡ Reward engineering
   - ➡ …
2. RL - core issues:
   - ➡ **Sample efficiency**
   - ➡ **Stability**
   - ➡ **Run time**
   - ➡ **Hyper-parameter tuning**
   - ➡ **Exploration**
   - ➡ **Safety**
   - ➡ **Robustness to Changes, Generalisation**
   - ➡ **…**

RL core issues

RL - core issues

- Sample efficiency
- Stability
- Run time
- Hyper-parameter tuning
- Exploration
- Safety
- Robustness to Changes
- Generalisation
- …

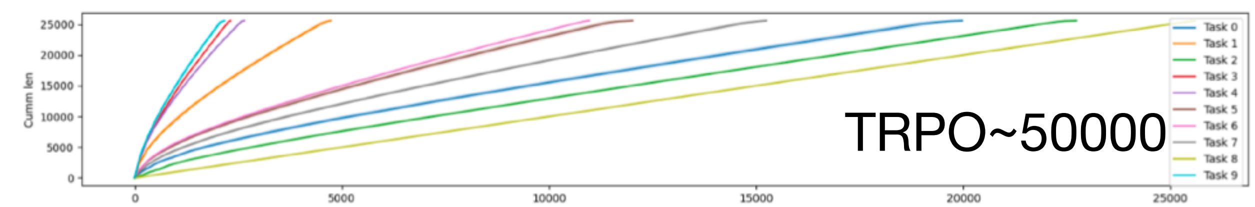
Sample efficiency

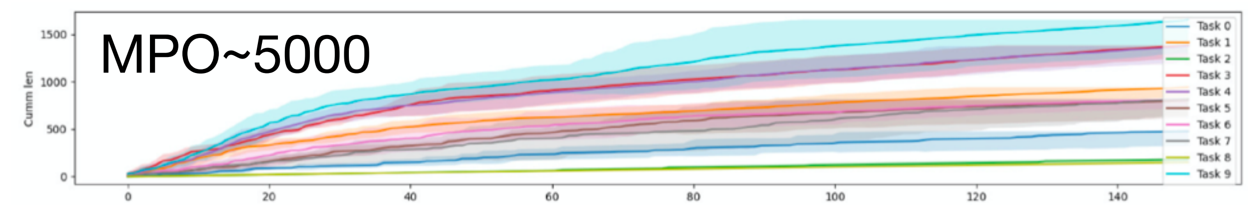
> 10e6 interactions

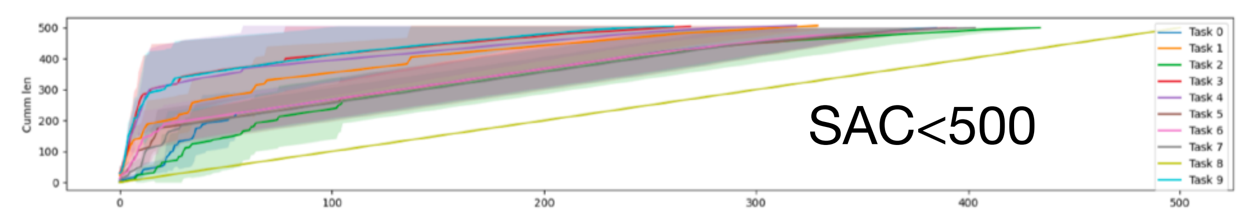
- Derivative free methods: (NES, CMA,..)
- 10 x Online methods (A3C)
- 10 x Policy-gradient methods (TRPO)
- 10 x Replay-Buffer + Value function estimation (Q-Learning, DDPG, TD3, NAF, SAC,…)
- 10 x Model-based RL methods (MPO, Guided Policy Search, Dyna)
- 10 x Model-based shallow methods (no NNs) Few shot GPs...

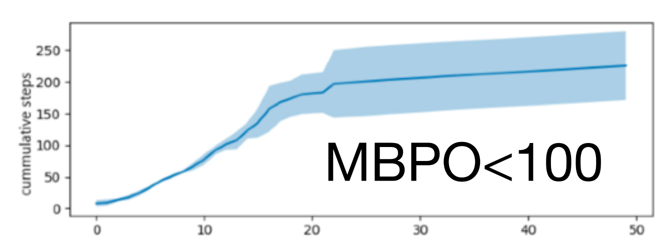
< 100 interactions

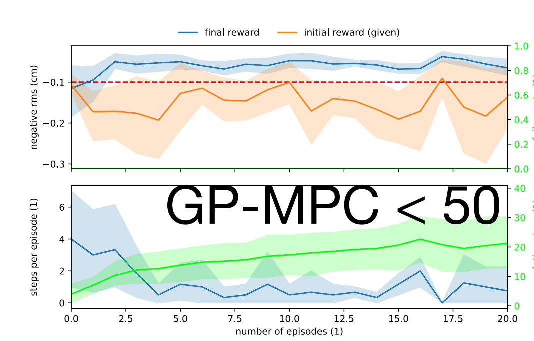
Colours are different tasks (optics)



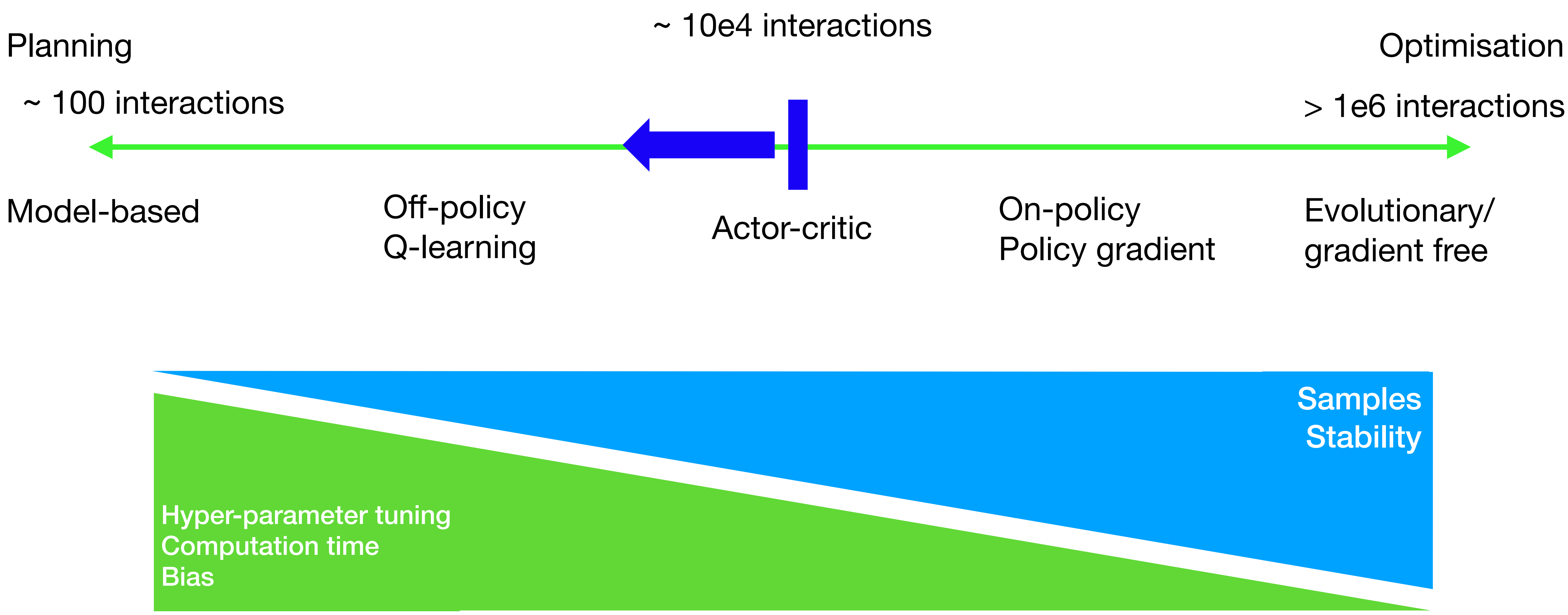




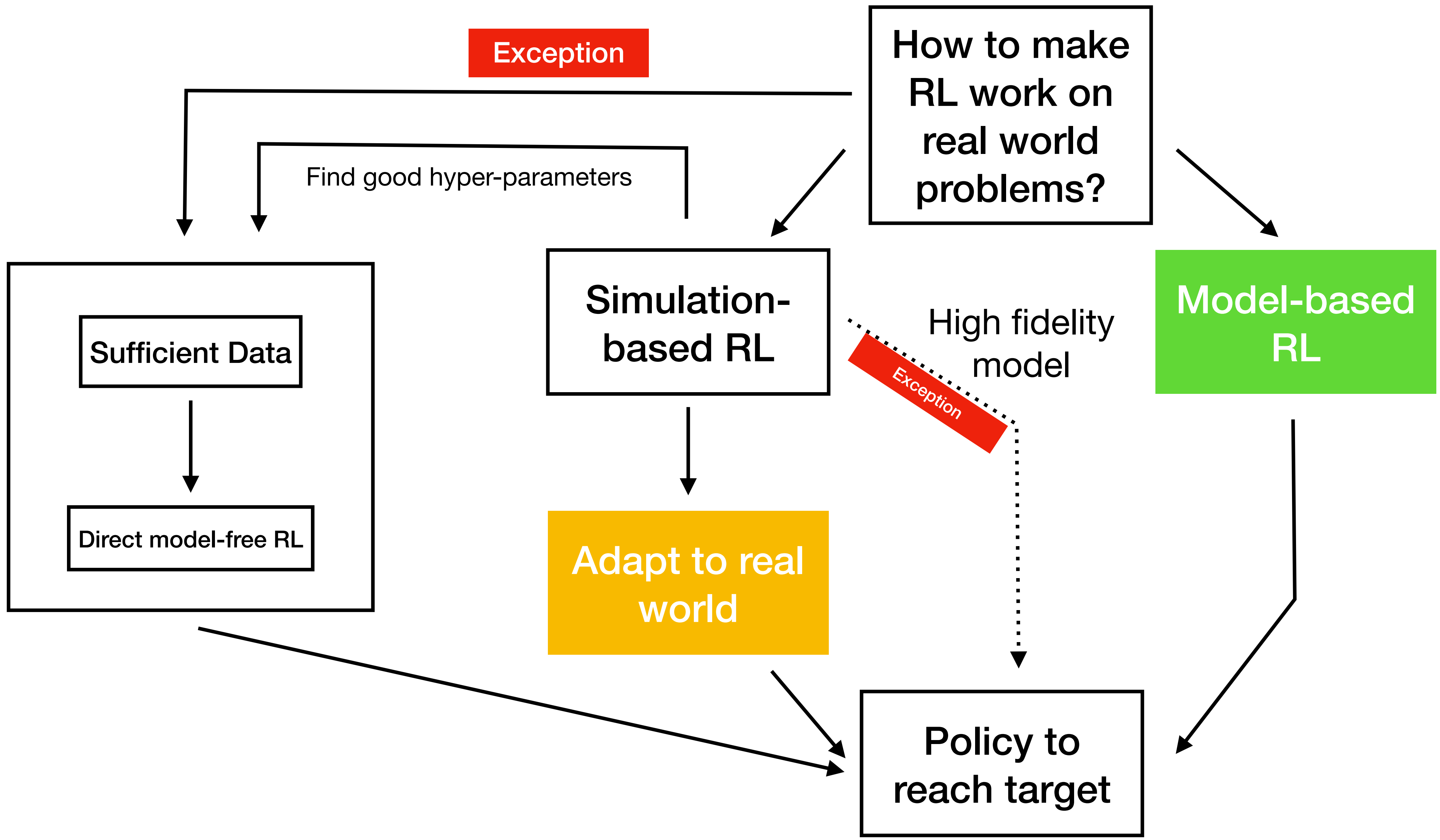




But sample efficiency is not all

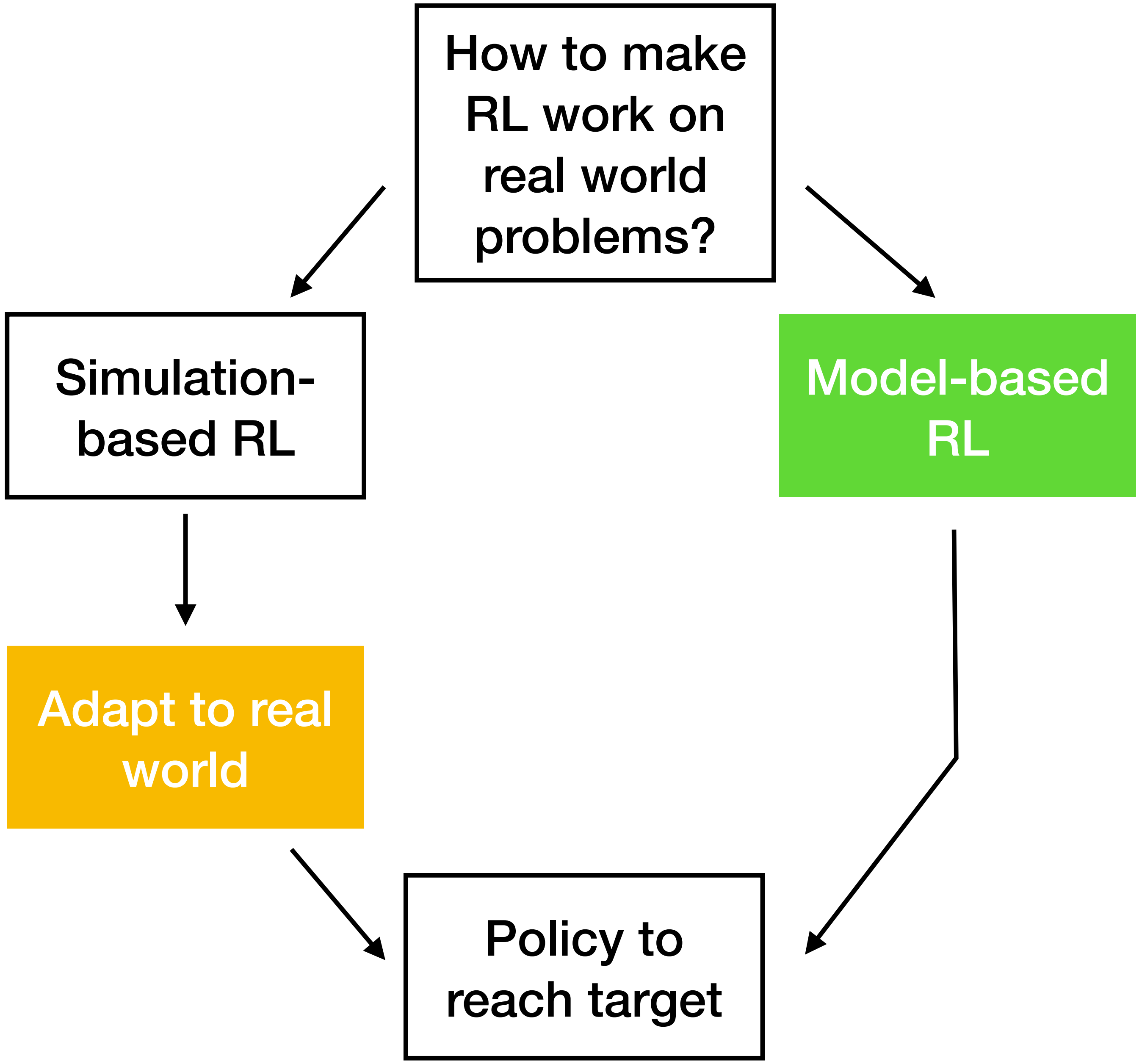


Scenarios RL2Real



Scenarios RL2Real

- If we have a simulation: prepare agent for real world training in the best way
- If we don't have a simulation: train the agent in the fastest best way, and other advantages
- If we trained an agent…
  - ➡ Works on one day but not on others…
- Agent should be able to adapt quickly



Two concepts at the boundary of RL

Part I - Meta RL

Simulation-based RL

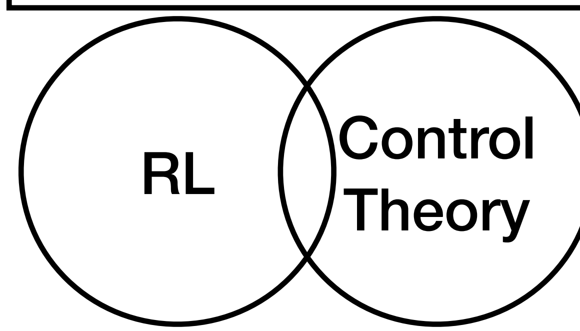
↓

Adapt to real world

- Meta RL
- Adapts quickly to changes
- Brings nice properties

Part II: safe shallow model-based RL

Direct RL



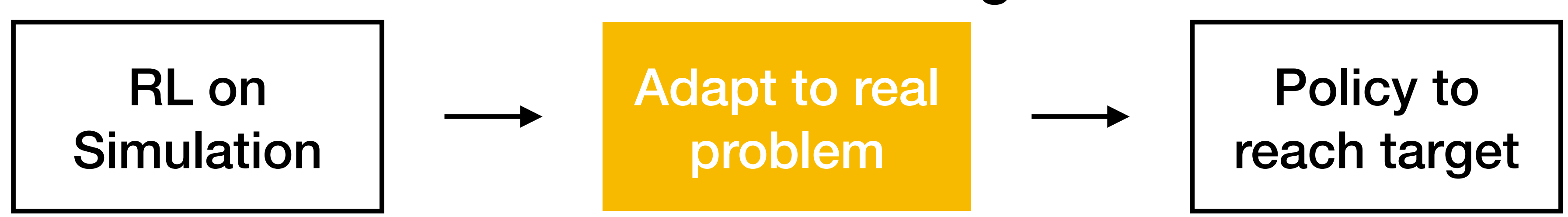
- RL towards control theory (leverage concepts from control theory)
- “The Bayesian optimisation of RL”
- Extremely sample efficient

Part I - Meta RL

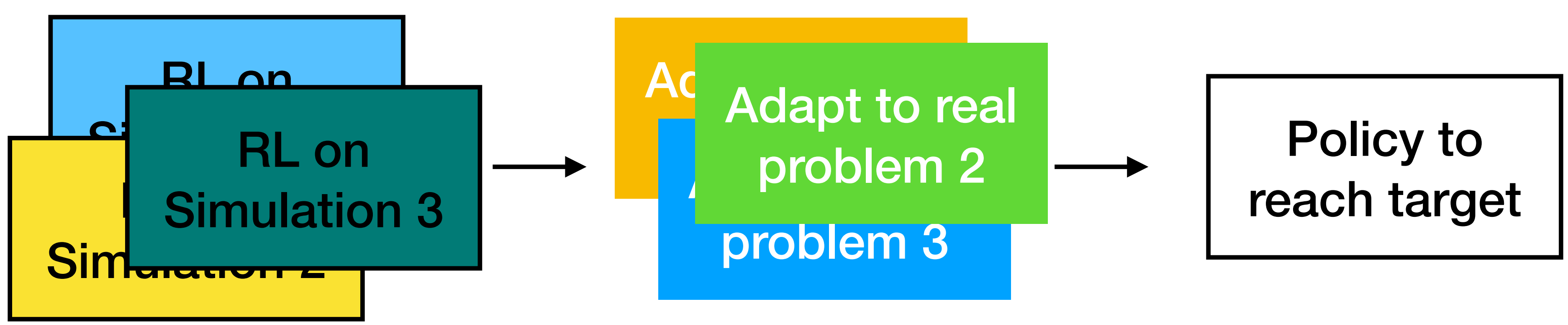
Motivation

- How we can use experience from some source domain to get into a position, where we can solve more efficiently or effectively new downstream tasks?
- Transfer learning: Using experience from one set of tasks for faster learning and better performance on a new task

Transfer learning



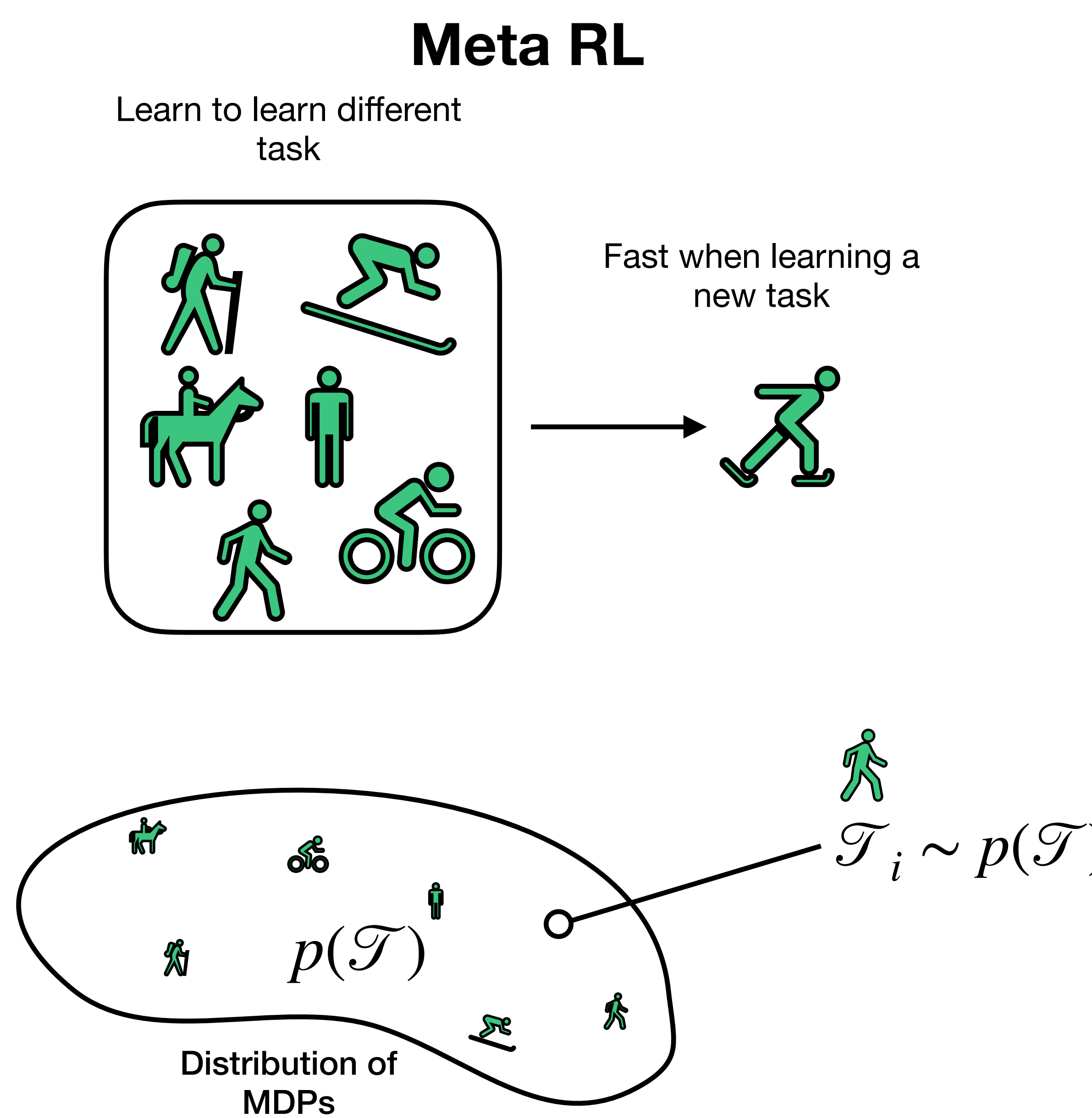
Domain randomisation/multi task learning



Can we do this smarter? → Meta-learning

What is meta-RL?

- Can the knowledge acquired from learning many different tasks be leveraged to <u>expedite and improve the learning</u> process for new tasks?
- Meta-learning = learn to learn
- Comes in many flavours - we focus on gradient based meta-learning
- Closely related to multi task learning- in multi-task is the task provided explicitly
- Meta-learning distinguishes itself by its ability to infer tasks and its <u>explicit focus on rapidly adapting to new task</u>



Model Agnostic Meta Learning (MAML)

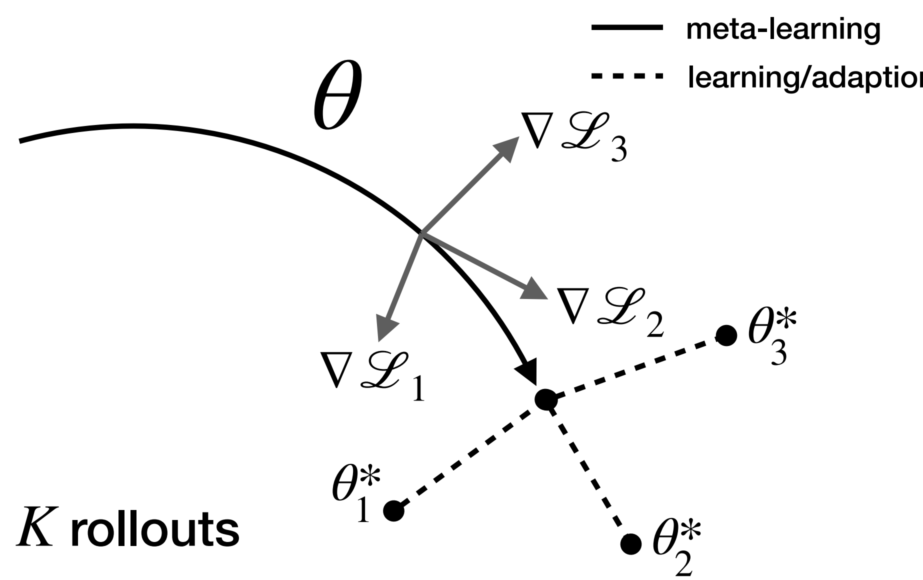
Why MAML is a good idea

- MAML is universally applicable beyond our specific scenario:
  - ➡ It can be implemented across various optimization problems.
  - ➡ The required gradients (to second order) can be efficiently computed using automatic differentiation.

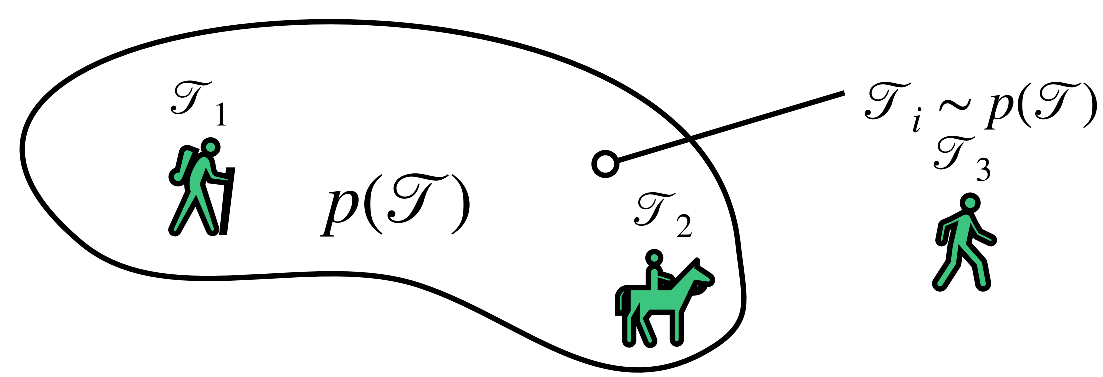
Meta RL via gradients

MAML outline

Require $p(\mathcal{T})$: distribution over tasks
Require α, β: step size hyper-parameters
1. randomly initialise θ
2. **while** not done **do**
3. sample batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
4. **for each** $\mathcal{T}_i$ **do**
5. Sample with policy θ: $\mathcal{D}^{tr}_{\mathcal{T}_i} \sim \mathcal{D}_{\mathcal{T}_i}$
6. Evaluate $\nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr}_{\mathcal{T}_i})$ with respect to K rollouts
7. Compute adapted parameters with gradient descent: $\phi_i = \theta - \alpha \nabla_\theta \mathcal{L}(\theta, \mathcal{D}^{tr}_{\mathcal{T}_i})$
8. Sample with new policy ϕ_i: $\mathcal{D}^{test}_{\mathcal{T}_i} \sim \mathcal{D}_{\mathcal{T}_i}$
9. Update $\theta \leftarrow \theta - \beta \nabla_\theta \sum_i \mathcal{L}(\phi_i, \mathcal{D}^{test}_{\mathcal{T}_i})$



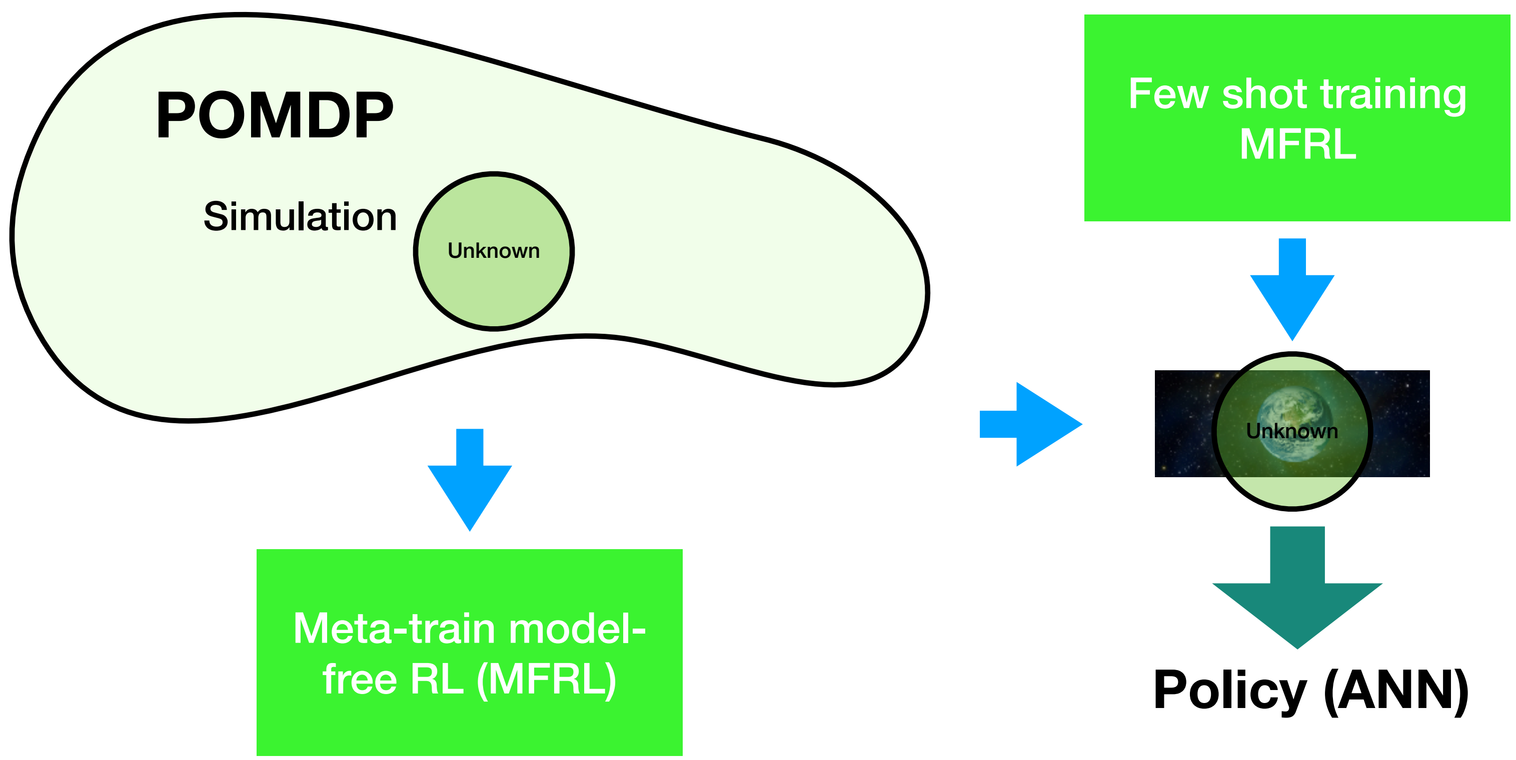
$$\mathcal{L}_{\mathcal{T}_i}(\theta) = -\mathbb{E}_{s_t, a_t \sim \pi_\theta, \mathcal{T}_i}\left[\sum_{t=1}^{H} R_i(s_t, a_t)\right]$$



Our set-up

- TRPO used for meta optimization
- Policy gradient with GAE (Schulmann 2015) as RL algorithm - fast and stable

Meta RL (in accelerator control)

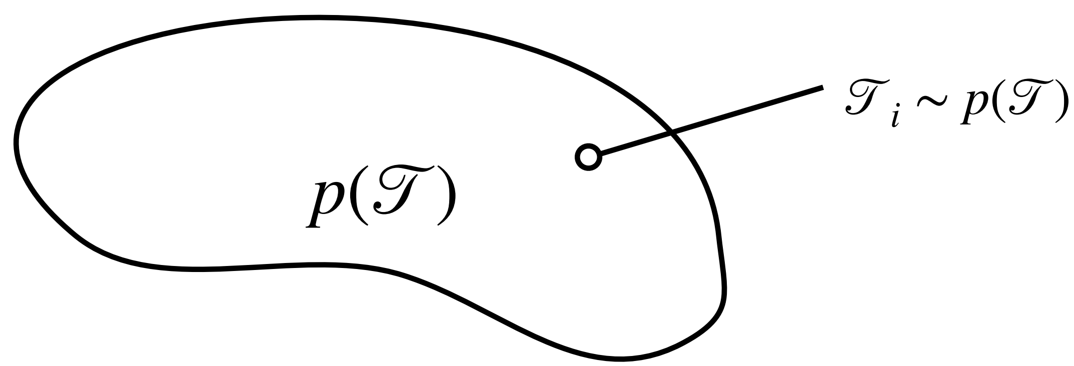


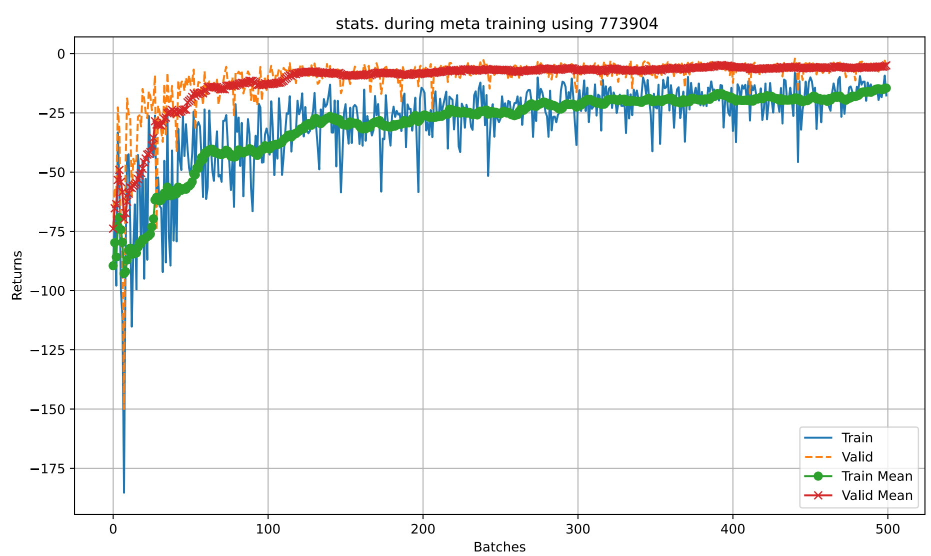
- Possible scenarios:
  - Inaccurate simulation → Prepare agent for real training in reliable and fast way
  - Non-stationarity → Environment changes regularly, fast, stable retraining
  - Several similar computational demanding problems → Common pre-training
  - …

Experiments

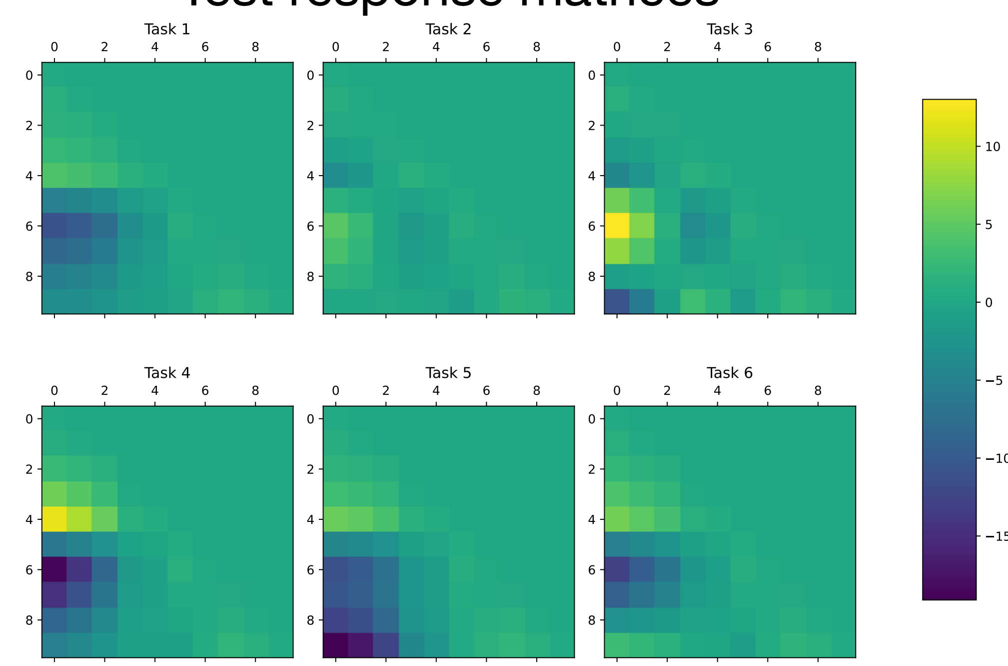
Our set-up

- Assume we don't know the optics (quadrupole settings) in advance
- Different optics are generated (quadrupoles are varied) within a uniform distribution centred on the real settings
- To assess progress, five optics, and the real optics, are fixed and progress is monitored



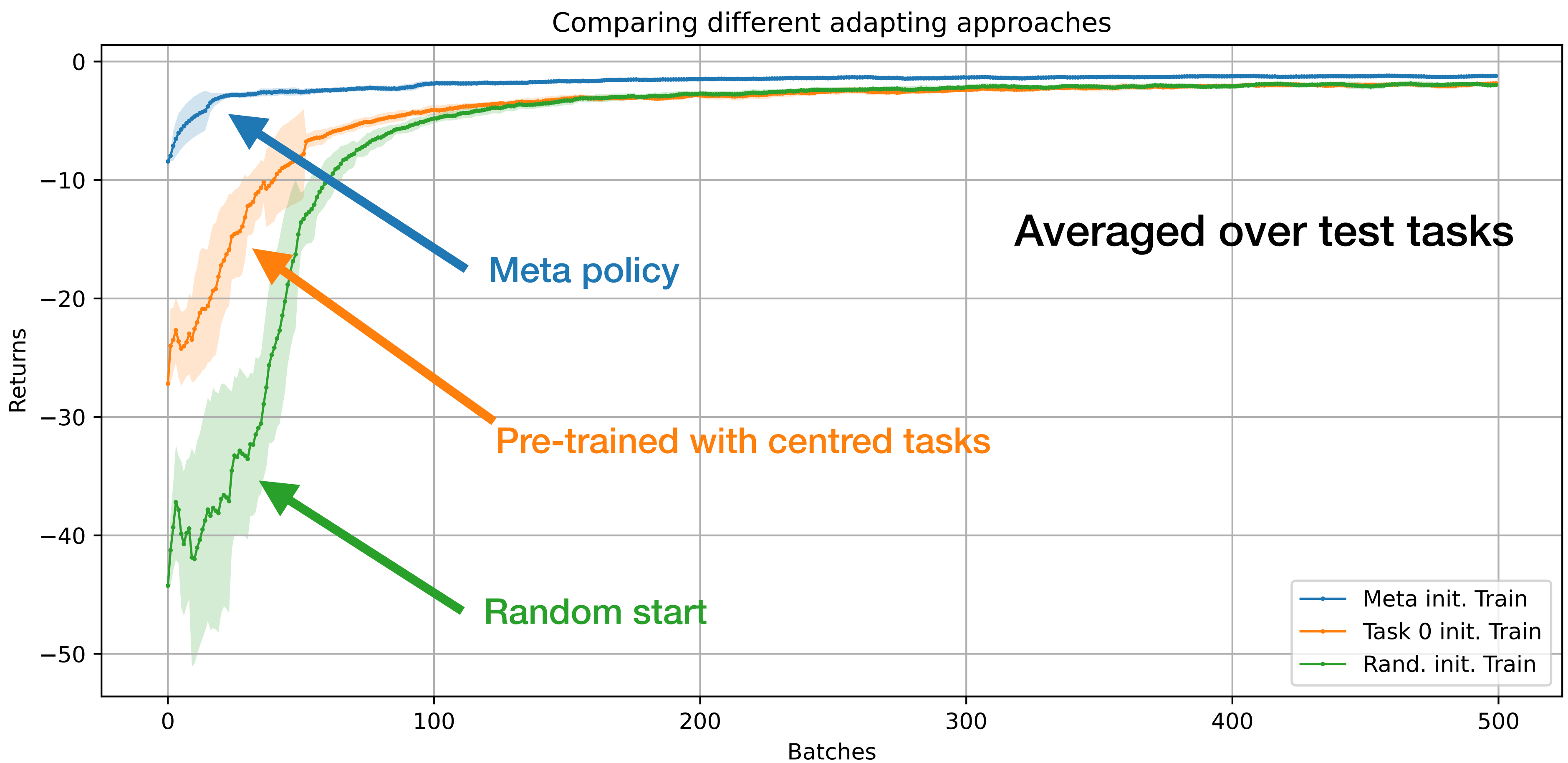


Test response matrices



Experiments Overview

- Stable and monotonic training from meta policy
- Quick adaption to actual setting - few shot adaption



Demonstrated on the machine with Lukas and Verena

Part II: safe shallow model-based RL

(Fast) RL with guarantees - a dream?

What if we'd know the model: optimal control

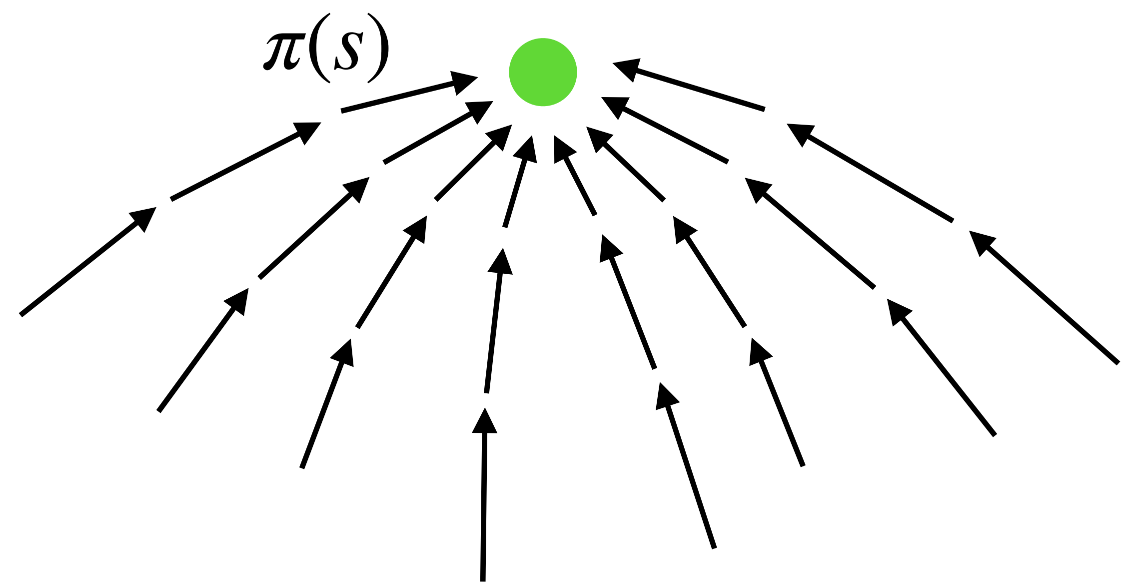
Three main ingredients

- Mathematical description of the system to be controlled (state-space models)
  - ➡ We use MDPs
- Specification of a performance criterion (the cost function)
  - ➡ The reward designed by us (or emitted by the environment in RL setting)
- Specification of constraints
  - ➡ Control or state constraints

Model assumptions

- Discrete time
- A stochastic dynamics with Markov property:
  $\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t, \omega_t)$ with $\omega_t = \omega_{t-1}(\mathbf{s}_t, \mathbf{a}_t)$
- Later ω_t is normally distributed
- In stochastic settings optimise for an expected reward

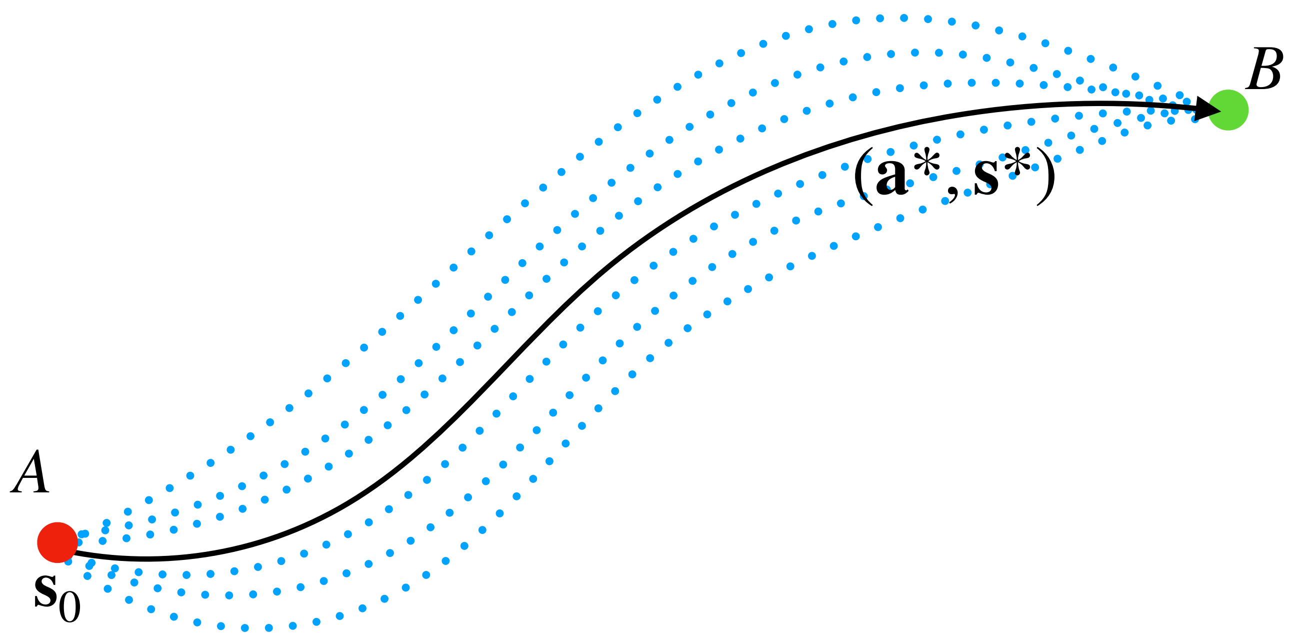
Solution 1: Dynamic programming



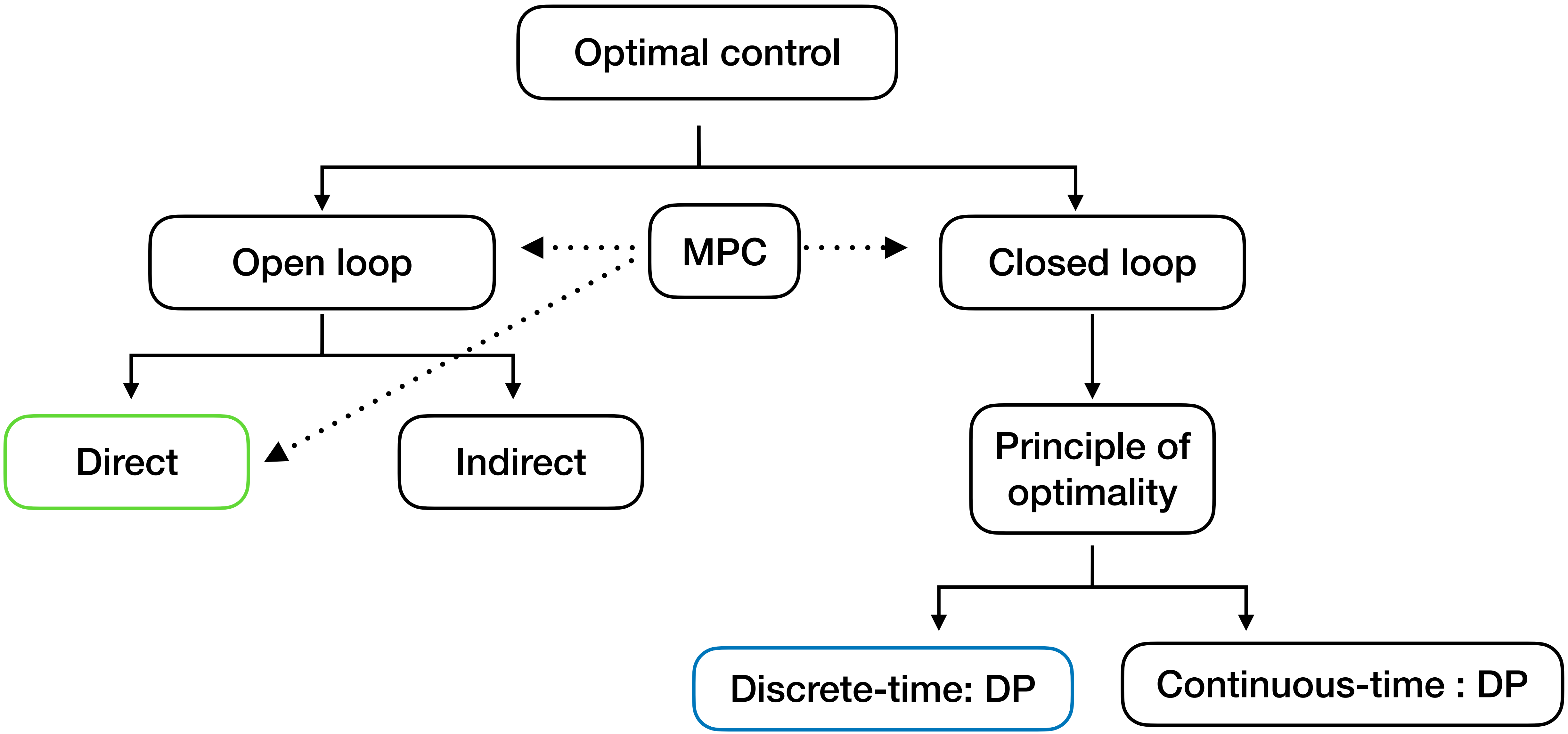
- Dynamic Programming (Principle of Optimality - sufficient condition)
  - ➡ Compositionality of optimal paths
  - ➡ Closed-loop solutions: find a solution for all states at all times
- Solvable via Bellman equation in a backward recursive fashion
- Algorithms as e.g. Value iteration, Policy iteration (see Sutton and Barto)
- <u>No direct notion of constraints for states or actions!</u>

Solution 2: Non-feedback control

- Calculus of Variations - Pontryagin Maximum Principle PMP (necessary condition)
- PMP turns functional minimisation in a function minimisation at each point in time
- Find a solution-sequence $(\mathbf{a}^*, \mathbf{s}^*)$ for a given initial state $\mathbf{s}_0$
- <u>Can handle constraints e.g. $\mathbf{s}_t \in S$, $\mathbf{a}_t \in A$</u>
- <u>But: open loop cannot stabilise the system!</u>

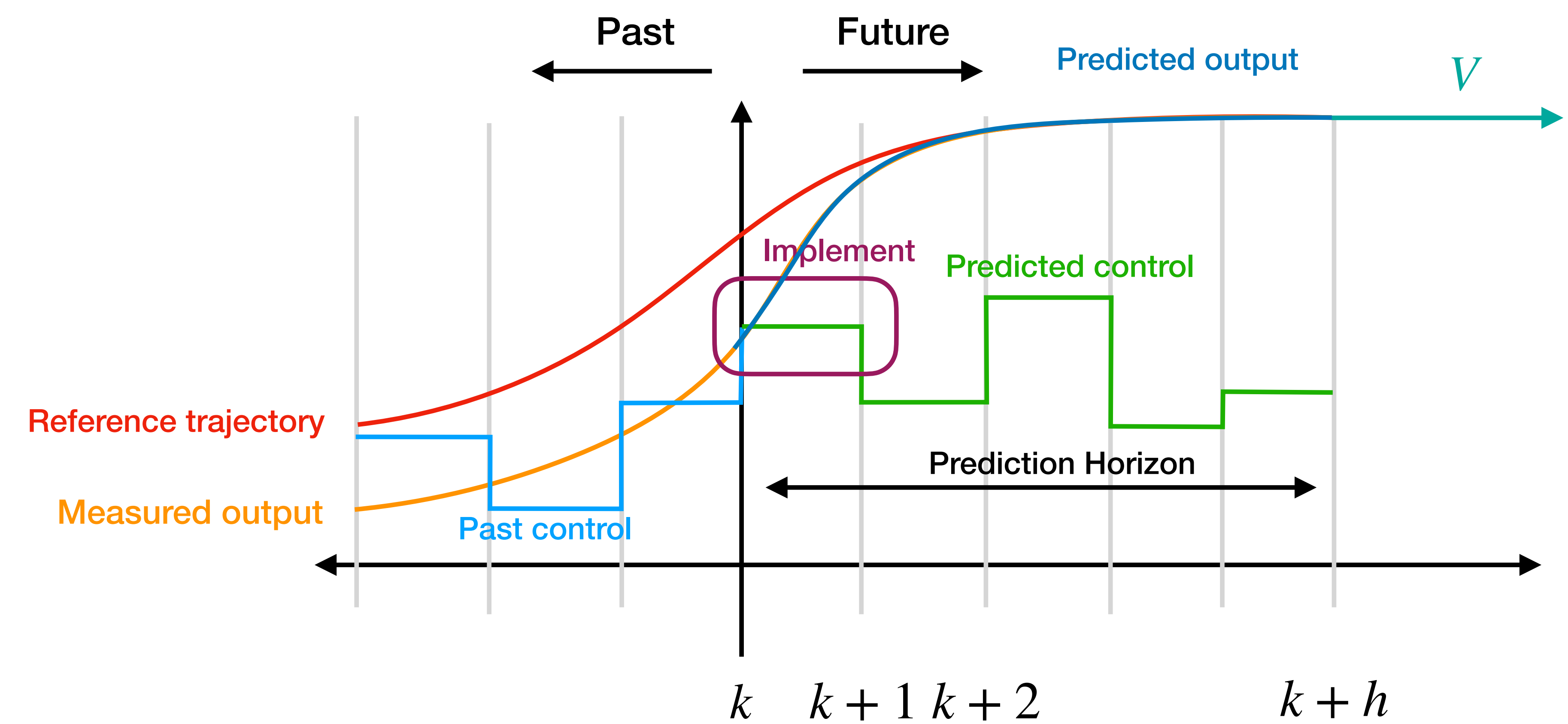


Best of both worlds - model predictive control (MPC)



Adapted from AA 203: Optimal and Learning-Based Control

MPC Idea



Want to solve infinite optimization problem:

$$\text{maximise}_{\pi_t} \lim_{T\to\infty} \mathbb{E}_{W_t}[\frac{1}{T}\sum_{t=0}^{T} R_t(S_t, A_t, W_t)]$$

subject to: $S_{t+1} = f_t(S_t, A_t, W_t)$

$A_t = \pi(S_t)$

$S_0 = s$

MPC computes an open loop control on finite horizon:

Optimise for finite horizon

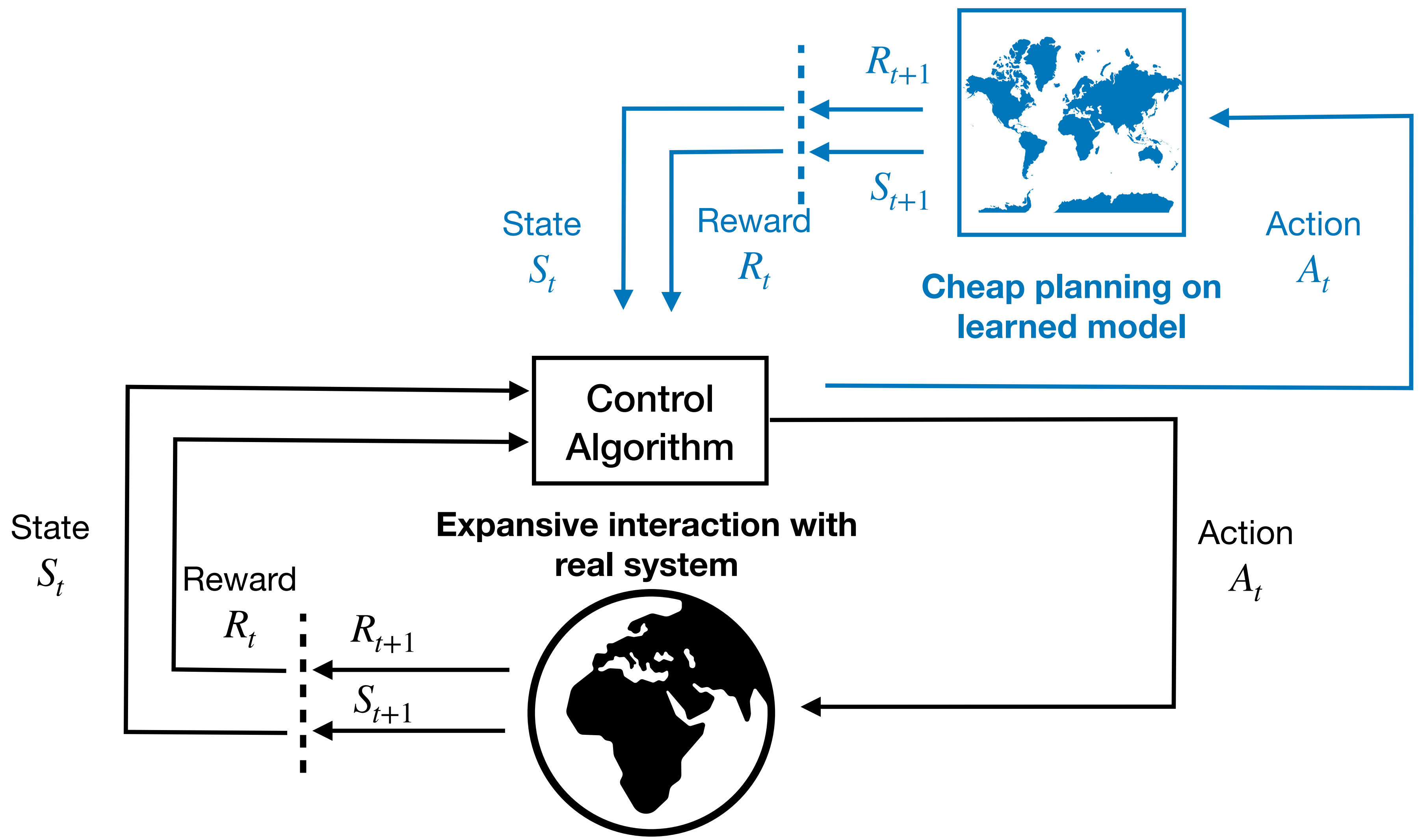
$$\text{maximise}_{\{a_t\}} \mathbb{E}_{W_t}[\boxed{\sum_{t=0}^{H-1} R_t(S_t, A_t, W_t)} + \boxed{V(S_H)}]$$

subject to: $S_{t+1} = f_t(S_t, A_t, W_t)$

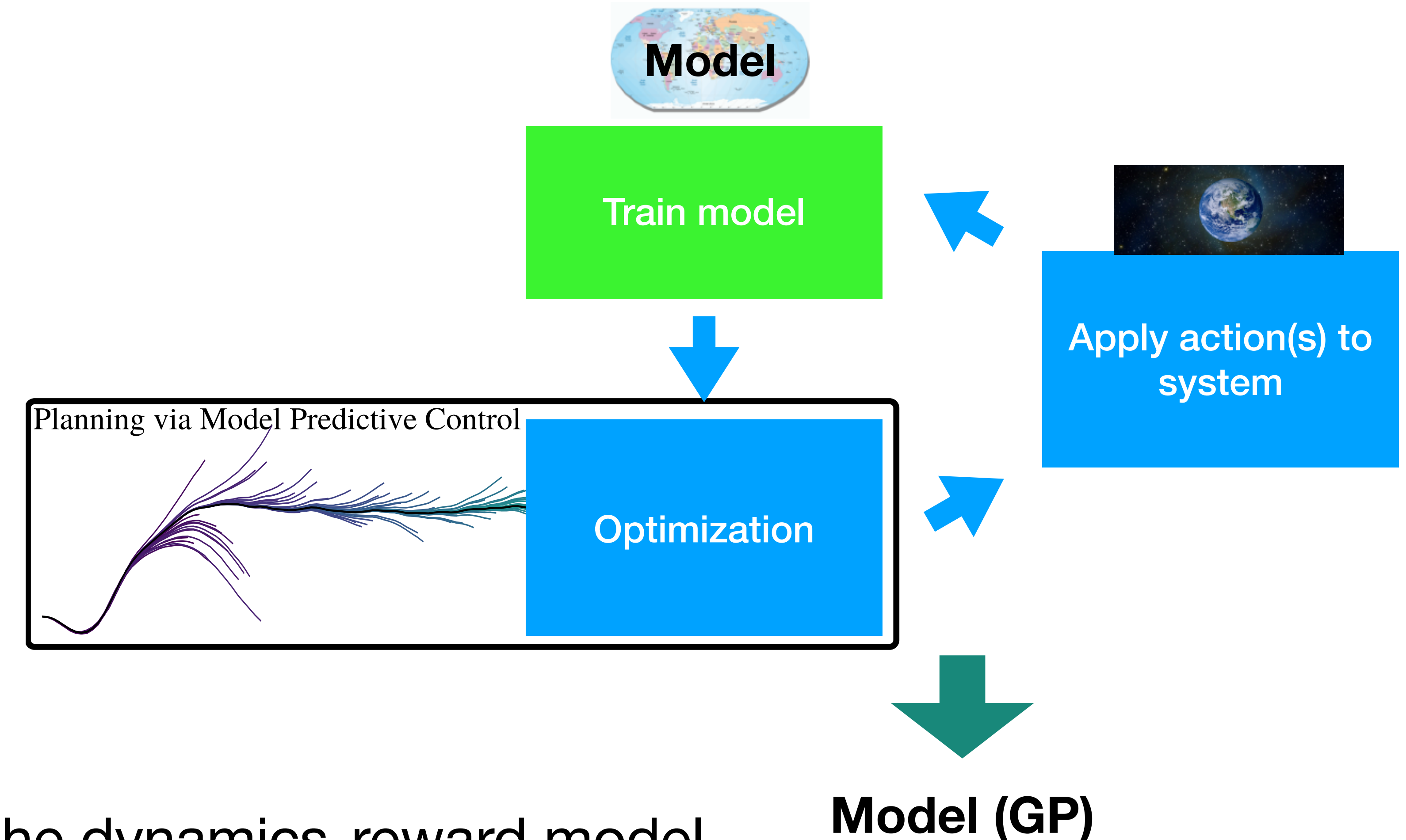
$S_0 = s$

Final cost performance for robustness

Back to RL - no model



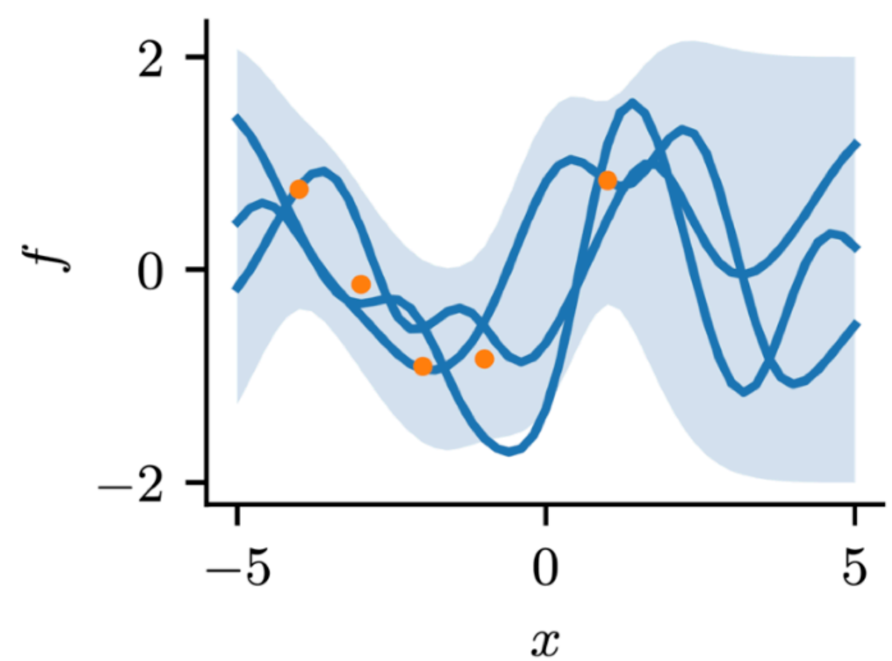
GP-MPC the BO of RL



- Setup the dynamics-reward model
- Use PMP to obtain sparse optimization with gradient information
- Choose optimization algorithm
- Consider safety (constraints)
- Set up training

We don't know the model

Example of GP



- Learn the model from data:
  - ➡ Aleatoric uncertainties
  - ➡ Epistemic uncertainties - minimise model bias
- Gaussian processes (GPs) are used assuming $\mathbf{s}_{t+1} = \mathbf{f}(\mathbf{s}_t, \mathbf{a}_t, \omega_t)$ and $\omega_t \sim \mathcal{N}(0,\sigma)$
- Include if needed the emitted reward
- Use RBF Kernel - allow for analytical propagation of uncertainties
- Standard GPs training: evidence maximization

Uncertainty propagation

- Moment matching for deterministic propagation of the mean $\mu(s_t)$ and the covariance $\Sigma(s_t)$ of the distribution of dynamics-reward model
- The immediate performance measure is: $\mathbb{E}[r(s_t, a_t)] = \int r(s_t, a_t)\mathcal{N}(s_t \,|\, \mu_t, \Sigma_t) ds_t$
- If reward not emitted - formulated as polynomial function

Fast optimisation

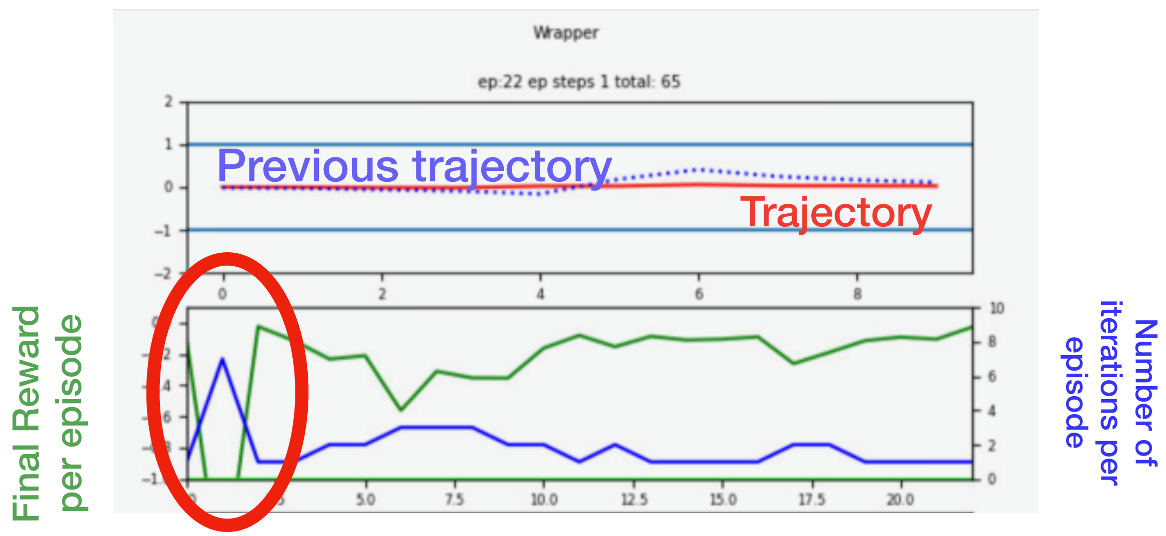
- From PMP a sequence of a constraint optimisation for each time step
- Dynamics-Lagrangian-multipliers in closed-form, Hamiltonian gradient same as Reward gradient
- Optimisation (analytical) up to (second) order in dynamics-reward model
- State and action constraints (analytical) up to second order
- “An interior point algorithm for large-scale nonlinear programming” - “trust-constr” used for experiments (we use BFGS)

Experiments

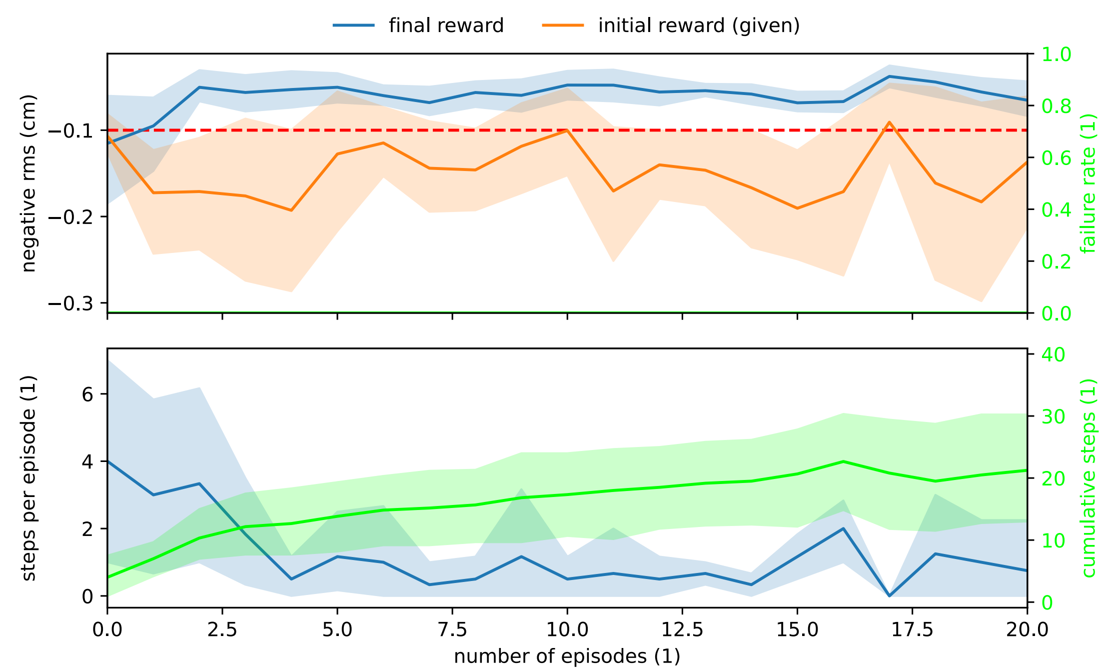
Tests on the machine - few shot RL

- November 2022 experiment campaign
- Adjusted on simulations
- Learns from scratch in a few steps
- Rapidly stabilises system

Screenshot during experiments



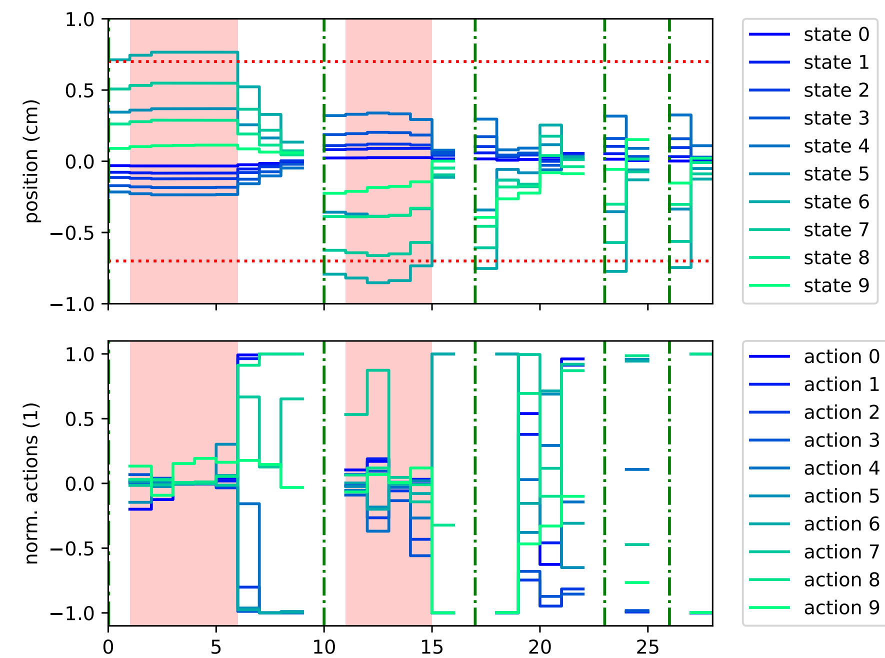
Average over all experiments



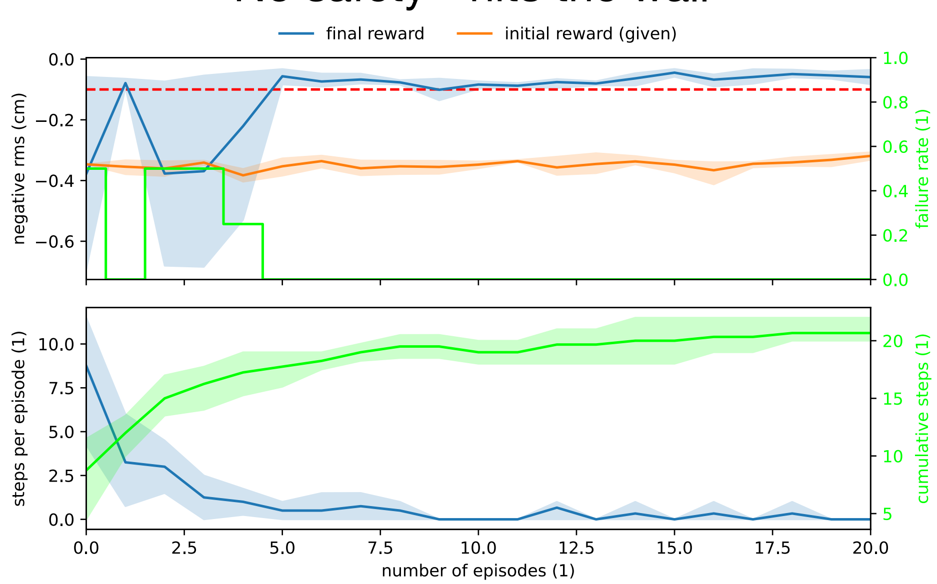
Incorporate considerations for safety

- Try to avoid hitting the wall
- Chance constrains: $\mathbb{P}(|s| > \text{threshold}) \geq \varepsilon \rightarrow$ safe policy is activated (red shaded)
- Two layer safety: longterm safety (for optimal control) and instant safety (for safe exploration)
- Initial settings close to wall to test safeness

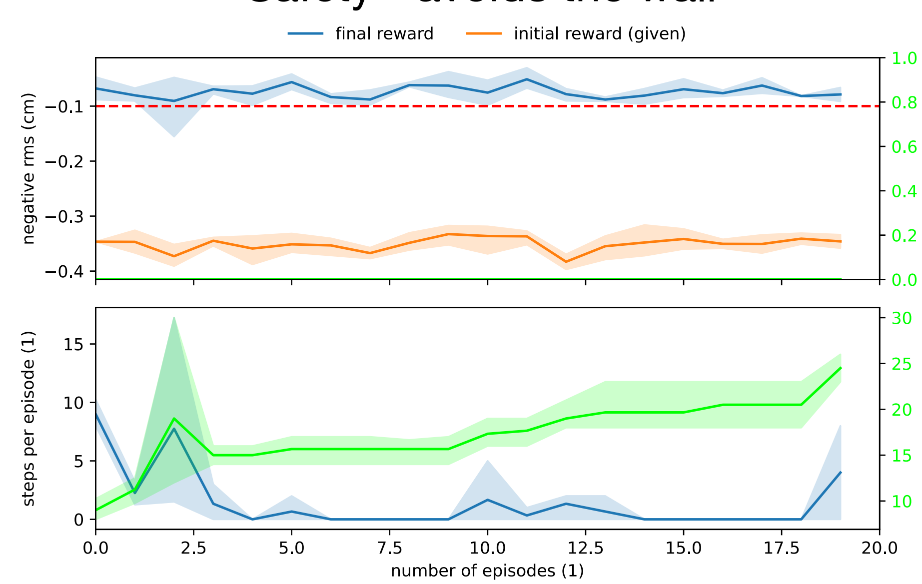
Safe exploration



No safety - hits the wall



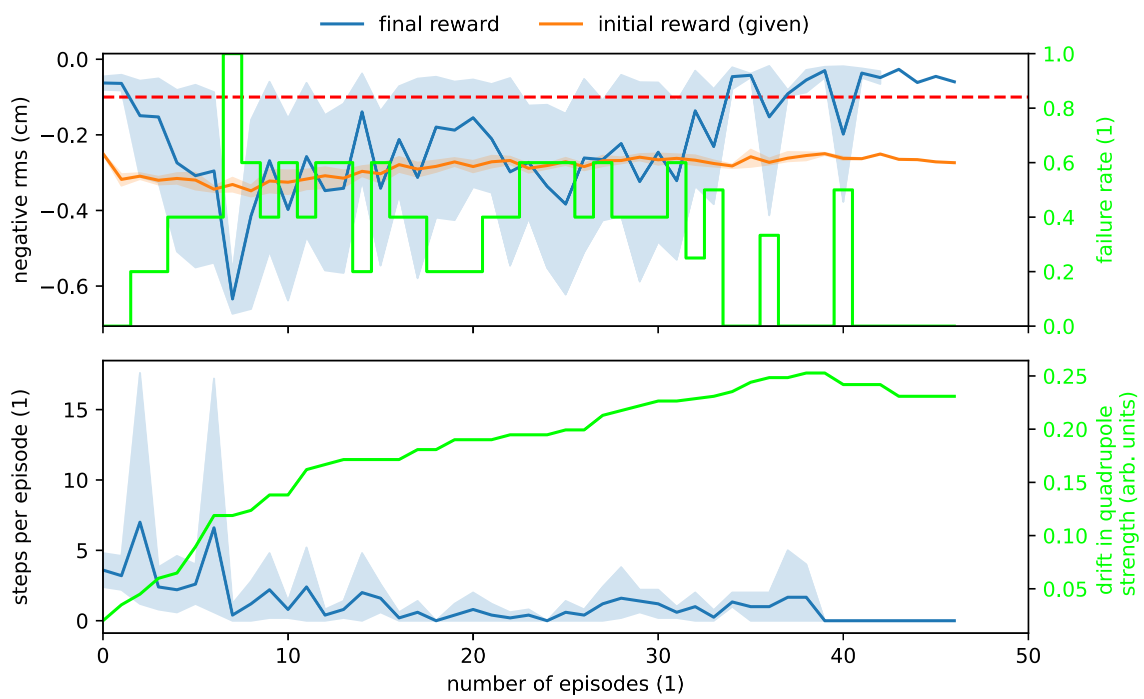
Safety - avoids the wall



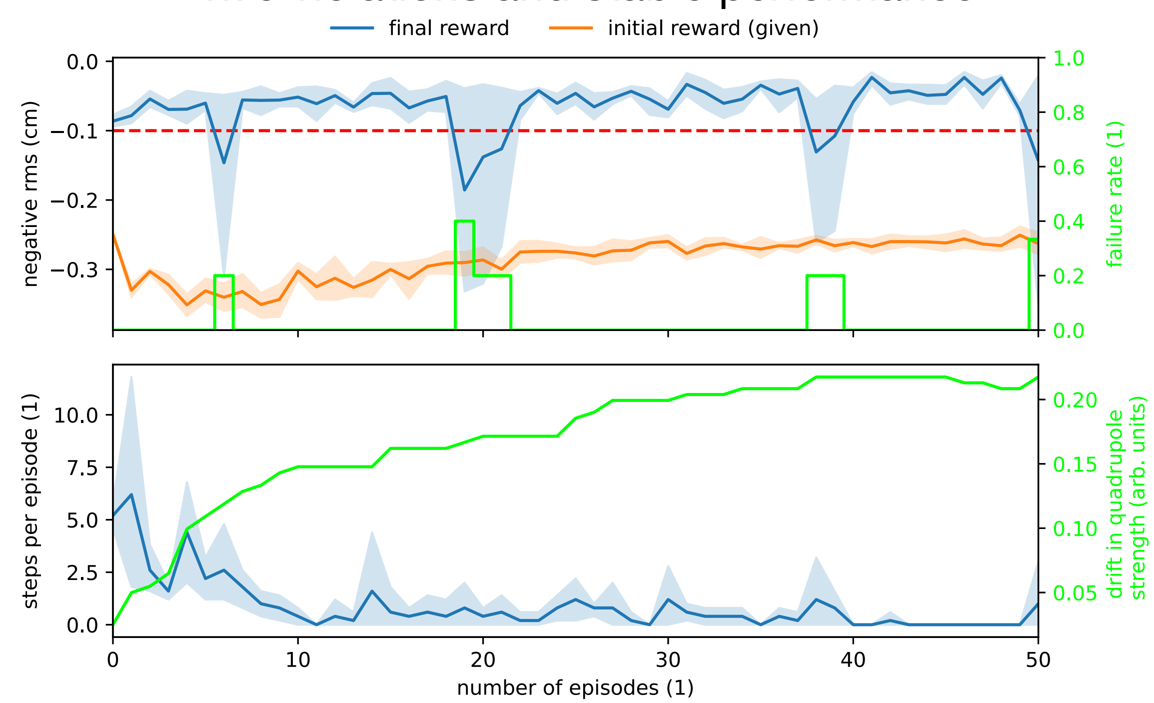
Non stationarity and safety

- Optics was distorted with a detuning of the quads by up to 20% with low timescale
- State was extended to incorporate the time step $s \rightarrow (s, t)$
- More weight on recent timepoints
- Safety also considered

No safety - no time information
no success in control



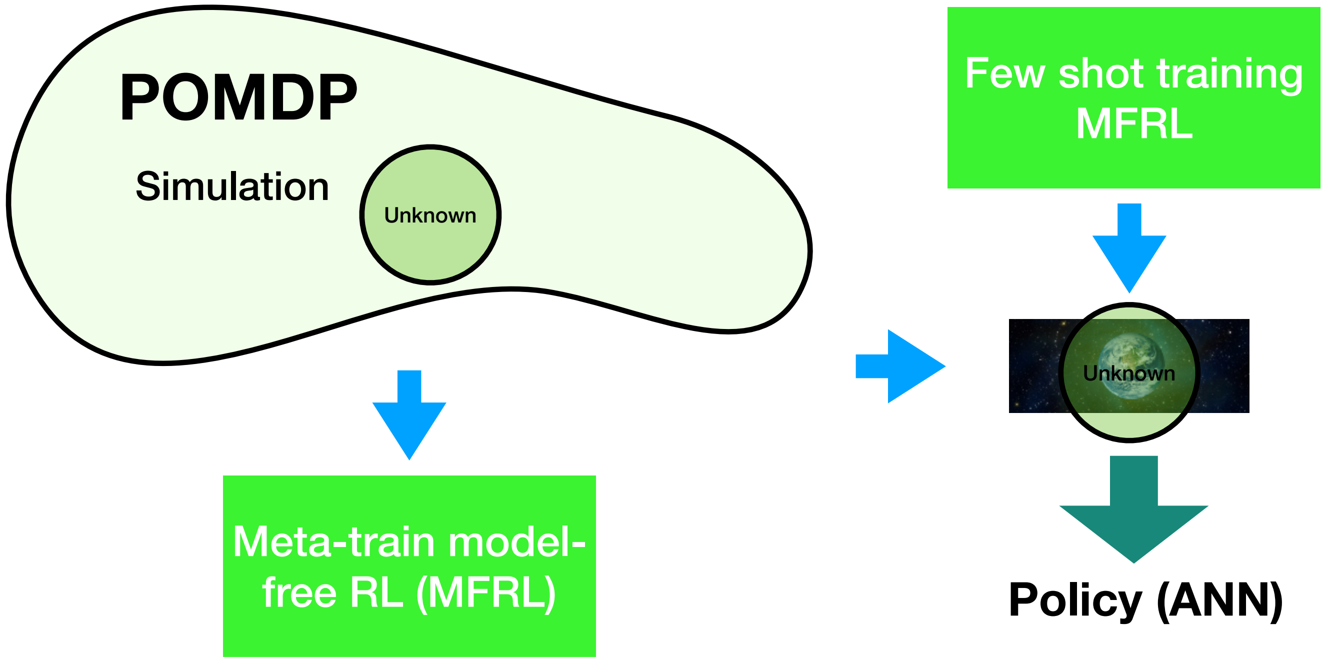
Safety + time information
little violations and stable performance



Wrap up

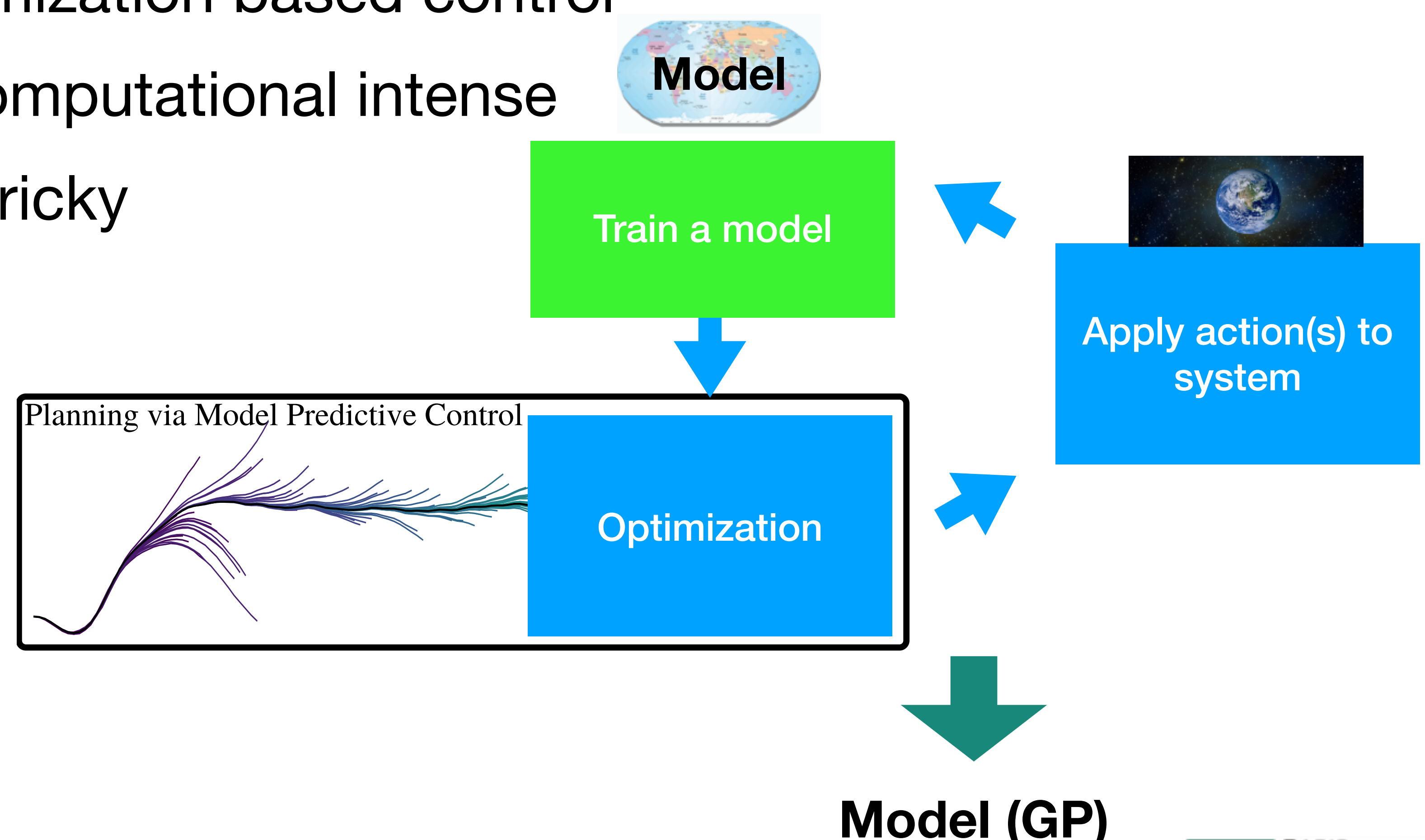
Key points - meta RL

- MAML leads to rapid and stable adaption, generalisation is good
- General simple and elegant concept (also applicable e.g. to BO)
- Stable and computationally fast and simple algorithms used (hardware)
- In the best case monotonic improvements during training (non destructive)
- Simulation needed covering the true problem as convex hull
- Meta training might be computational intense
- Implementation might be tricky
- Tuning is hard



Key points - GP-MPC

- Extremely sample efficient
- Can handle constrains
- GP is non-parametric → computational intense, scales badly
- Only model is stored, optimization based control
- Long horizons might be computational intense
- Implementation might be tricky
- Tuning is hard



Summary

- Machine learning is always a trade-off between several criteria (no free lunch) - the more tools the better
- The unique characteristics of the accelerator domain and real-world limitations narrow down the range of methods available, making the implementation of reinforcement learning a complex task
- Two RL methods are showcased to guide new research and ultimately achieve operational RL

Thanks for your attention

Now let’s have fun

References

- Courses online:
  - Chelsea Finn (Berkley): Deep Multi-Task and Meta Learning
  - Sergey Levine (Berkley): Deep Reinforcement Learning
  - Emma Brunskill (Stanford): Reinforcement Learning
- Papers:

  - C. Finn, P. Abbeel, and S. Levine, “Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks.” 2017: https://arxiv.org/abs/1703.03400
  - S. Kamthe and M. Deisenroth, “Data-Efficient Reinforcement Learning with Probabilistic Model Predictive Control,” in *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, A. Storkey and F. Perez-Cruz, Eds., in Proceedings of Machine Learning Research, vol. 84. PMLR, 2018, pp. 1701–1710. [Online]. Available: https://proceedings.mlr.press/v84/kamthe18a.html
  - A. Girard, C. E. Rasmussen, J. Quinonero- Candela, and R. Murray-Smith. Gaussian Process Priors with Uncertain Inputs-Application to Multiple-Step Ahead Time Series Forecasting. Ad- vances in Neural Information Processing Systems, 2003.
  - S. Hirlaender, L. Lamminger, G. Zevi Della Porta, and V. Kain, “Ultra fast reinforcement learning demonstrated at CERN AWAKE,” JACoW IPAC, vol. 2023, p. THPL038, 2023, doi: 10.18429/JACoW-IPAC2023-THPL038 https://cds.cern.ch/record/2886522/files/document.pdf
  - J. Beck et al., “A Survey of Meta-Reinforcement Learning.” 2023: https://arxiv.org/abs/2301.08028
  - J. Schulman, S. Levine, P. Moritz, M. I. Jordan, and P. Abbeel, “Trust Region Policy Optimization.” 2017: https://arxiv.org/abs/1502.05477
  - J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, “High-Dimensional Continuous Control Using Generalized Advantage Estimation.” 2018: https://arxiv.org/abs/1506.02438

- Books:

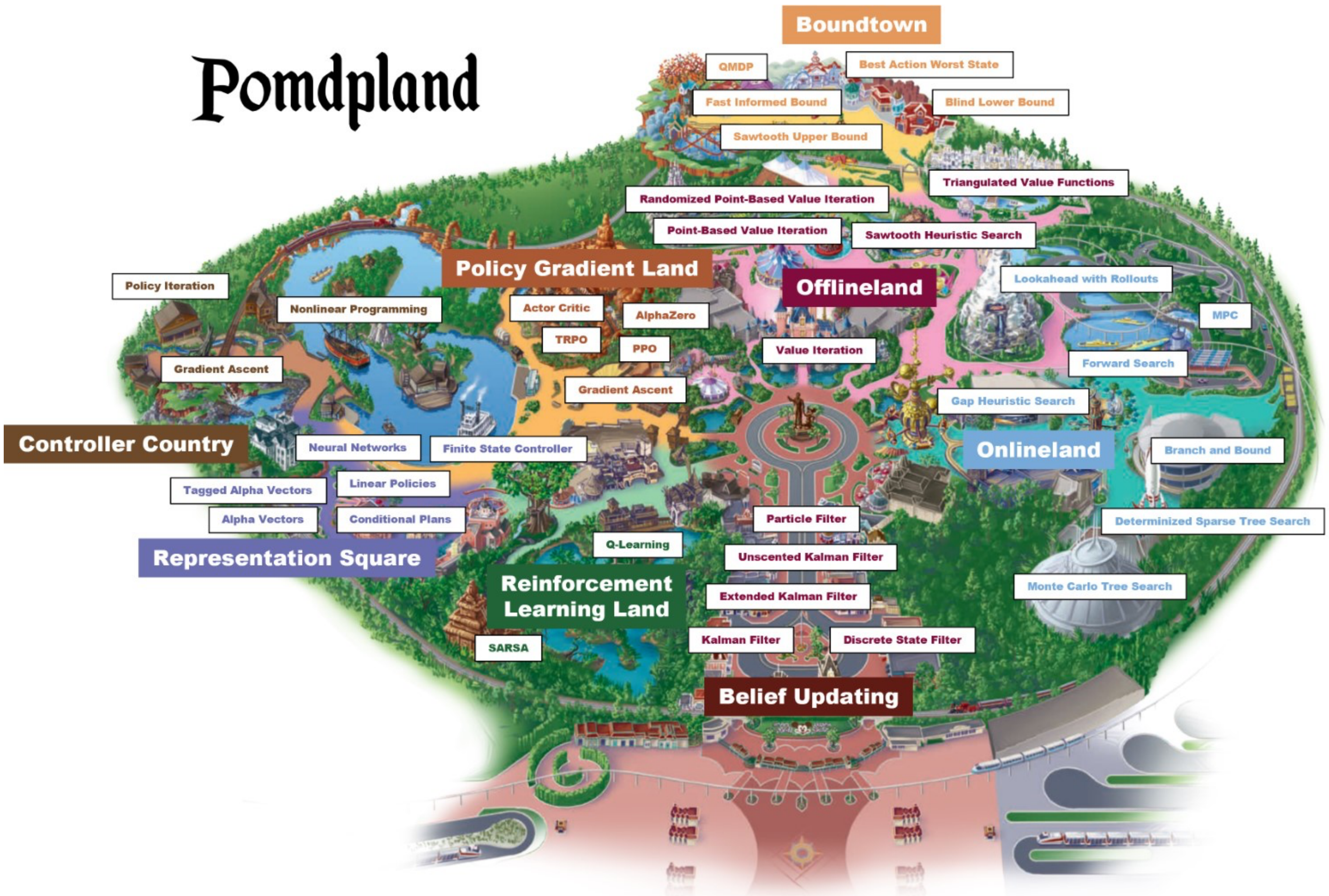
  - R. S. Sutton, *Reinforcement learning*, Second edition. in Adaptive computation and machine learning. Cambridge, Massachusetts: The MIT Press, 2020 http://incompleteideas.net/book/the-book.html
  - A. Agarwal, N. Jiang, S. M. Kakade, W. Sun: *Reinforcement Learning: Theory and Algorithms, 2022* https://rltheorybook.github.io/
  - K. P. Murphy, *Probabilistic Machine Learning: An introduction*. MIT Press, 2022. https://probml.github.io/pml-book/book1.html
  - K. P. Murphy, *Probabilistic Machine Learning: Advanced Topics.* MIT Press, 2023. http://probml.github.io/book2
  - D. Liberzon, *Calculus of variations and optimal control theory.* Princeton, NJ: Princeton University Pres, 2012, http://liberzon.csl.illinois.edu/teaching/cvoc/cvoc.html
  - F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control for Linear and Hybrid Systems*. Cambridge: Cambridge University Press, 2017

- Other resources:

  - S. Hirlaender, Advanced concepts in RL, 2023 (RL4AA23 Lecture) https://github.com/RL4AA/RL4AA23/blob/main/slides/Hirlaender_advanced_concepts.pdf
  - S. Boyd, Convex Optimization: https://web.stanford.edu/class/ee364b/lectures.html

Problem formulation - capturing the problem in an MDP

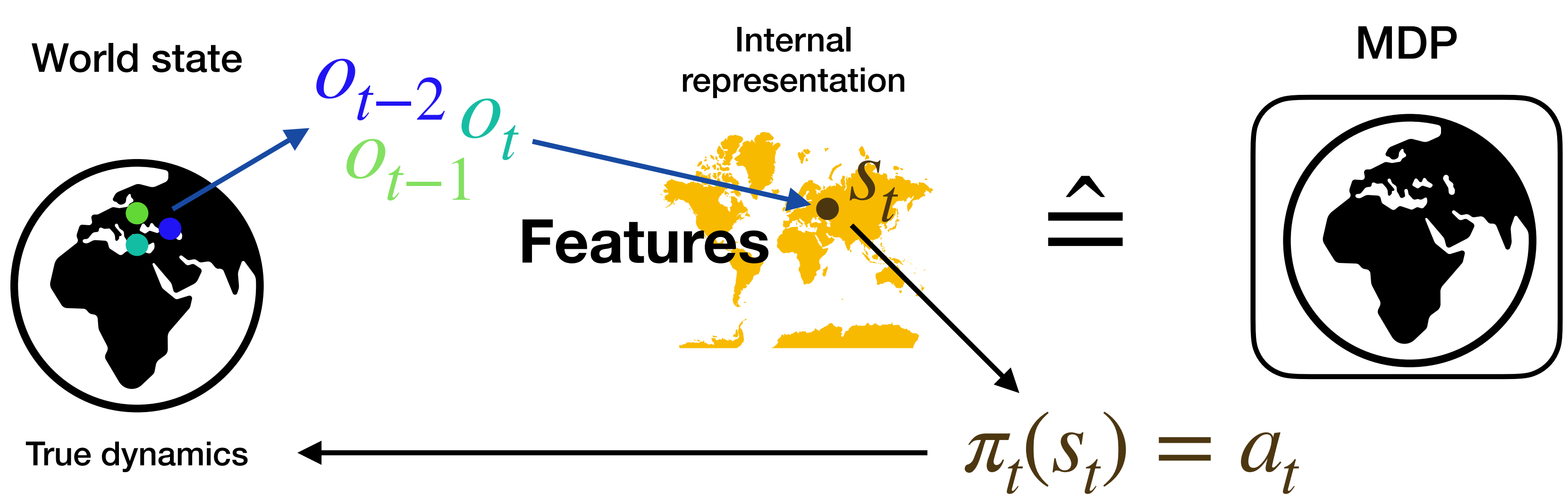
Wellcome to POMDPs



From Mykel Kochenderfer

Problem design - capture the right thing

- Solve an SDM problem: Information→Decision→Information→Decision→…
- Generally stochastic!
- Consequently we build a feedback system not planing too far in the future:
  - Define a **state** $s_t = h_t(o_t, a_{t-1}, o_{t-1}, a_{t-2}, o_{t-2} \ldots)$, as a function holding **sufficient statistics** until time step t for a decision - (example pong)
  - Decision based on s_t via: $a_t = \pi_t(s_t)$ - the policy - optimise an expected aggregate of future rewards



- Rarely the observation o is the state s, the world state is, but often we assume it is certainty equivalence!
- POMDP ⇒ MDPs!

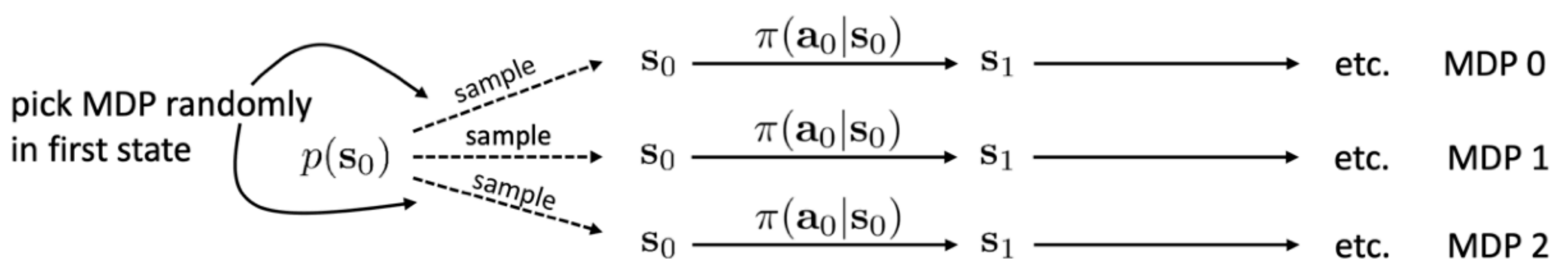
How bad is it?

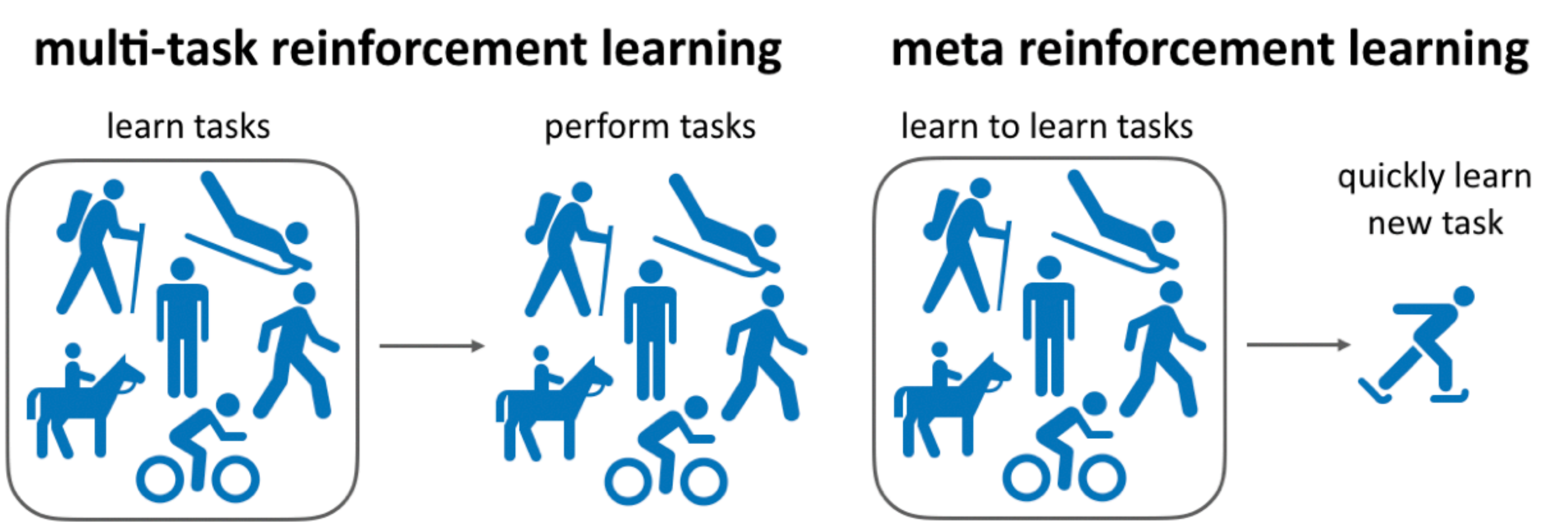
- Linear POMDP: believe state - $O_t = h_t(S_t, A_t, W_t)$
  - ➡ Static output feedback is NP hard (linear in O_t and dynamics)
  - ➡ General POMDPs are PSPACE hard
- There are ways out - separation principle:
  - ➡ Filtering $\hat{s}_t = f(\{o_t\})$ - prediction problem
  - ➡ Action based on certainty equivalence
  - ➡ Optimal filtering - if dynamics are linear and noise is Gaussian - Kalman filtering - general belief propagation - LQG
  - ➡ Kalman filtered state - optimal in estimation and control
  - ➡ Estimate state with prediction $S_t = h(\tau_t)$, τ_t are time lags

POMDPs and non stationarity

- To find a proper state we have to solve the additional prediction problem $s_t = h_t(o_t, a_{t-1}, o_{t-1}, a_{t-2}, o_{t-2} \dots)$
- In the non-stationary, finite horizon formulation the MDP has the form $(S, A, \{P\}_h, \{r\}_h, H, \rho_0) \Rightarrow$ Value-functions $Q_h(s, a)$ get time depended $\Rightarrow$ similar form of Bellman equations
- We can incorporate time into state e.g. $\tilde{s} = (s, h) \Rightarrow$ standard MDP
- Generally Bellman equation nice in discounted, stationary formulation $\Rightarrow$ this is what we usually see and most libraries build on this formulation

Multi task vs meta RL





Direct policy search

- RL as derivative free optimization:
    - ➡ maximise$_{z \in \mathbb{R}^d} R(z) \Rightarrow$ maximise$_{p(z)} \mathbb{E}_p[R(z)]$
    - ➡ Parametrise a distribution $p(z;\theta) \Rightarrow$ maximise$_{p(\theta)} \mathbb{E}_{p(z;\theta)}[R(z)]$
    - ➡ Likelihood trick - estimate the derivative:

$$\nabla_\theta J(\theta) = \int R(z) \nabla_\theta p(z;\theta) dz = \int R(z) \frac{\nabla_\theta p(z;\theta)}{p(z;\theta)} p(z \; \theta) dz$$

- $$= \int R(z) \nabla_\theta \log p(z;\theta) p(z \; \theta) dz = \mathbb{E}_{p(z;\theta)}[R(z) \underbrace{\boxed{\nabla_\theta \log p(z;\theta)}}_{\text{Score function}}]$$

- Unbiased gradient estimate of J, if sample efficiently from $p(z;\theta)$ and $\log p(z;\theta)$
- High variance

Probabilistic trajectories

- Objective if episodic: $J(\theta) = V^{\pi_\theta}(s_0) := V(\theta)$
  - ➡ Stochastic search: pure random search, Simplex, Bayesian optimization
- Using the gradient:

$$V^{\pi}(s_0) = \mathbb{E}_\pi[\sum_t \gamma^t R_{t+1} \,|\, S_t = s_0]$$

Trajectory probability

➡ $$V(\theta) = \sum_\tau \boxed{P(\tau;\theta)}\boxed{R(\tau)}$$

Trajectory reward

➡ $$\nabla_\theta V(\theta) = \sum_\tau P(\tau;\theta)R(\tau)\boxed{\nabla_\theta \log P(\tau;\theta)} = \mathbb{E}[R(\tau)\,\nabla_\theta \log P(\tau;\theta)]$$

Log likelihood trick Stochastic gradient

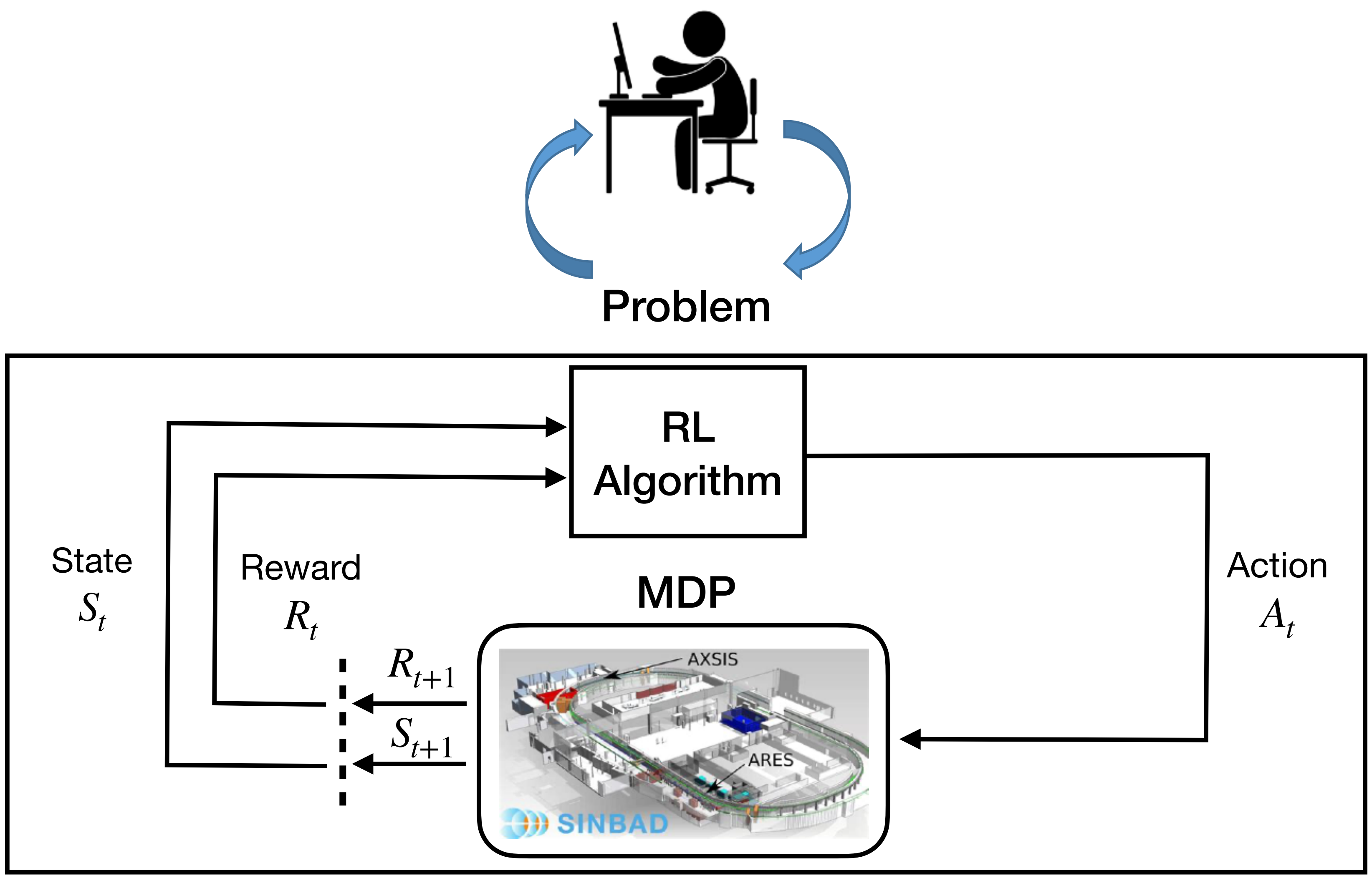
➡ Sampling of $A_t \sim p(\cdot \,|\, \tau_t;\theta)$

- Handle probabilistic policies (example)
- High dimensional and continuous action spaces
- Reinforce algorithm considers temporal structure

➡ Finite difference approximation ≙ Reinforce algorithm

The entire problem

Markov decision process - MDP



Optimisation

- Optimisation has become a standard tool in the control room:
  - ➡ Fast adaption from scratch
  - ➡ Easy to tune with short exploration
  - ➡ It is not RL - optimisation is greedy
- RL has potential to solve a much broader range of problems:
  - ➡ Incorporates state information - if trained, much faster than optimization
  - ➡ Can handle delayed consequences
  - ➡ Policy might be faster and easier to calculate and implement

Wishlist

- An agent which is:
  - ➡ Easy to train
  - ➡ Needs little amount of samples or adapts from uncertain simulation
  - ➡ Adapts quickly or continuously to changes
  - ➡ Does not consume to much resources
  - ➡ Generalises well
  - ➡ Respects safety