A few reinforcement learning stories

Optimal Decision Making for Complex Problems

R. Fonteneau

University of Liège, Belgium

March 7th, 2018

Outline

Context: machine learning & (deep) reinforcement learning in brief

Batch Mode Reinforcement Learning

Synthesizing Artificial Trajectories

Estimating the Performances of Policies

Context: machine learning and (deep) reinforcement learning in brief

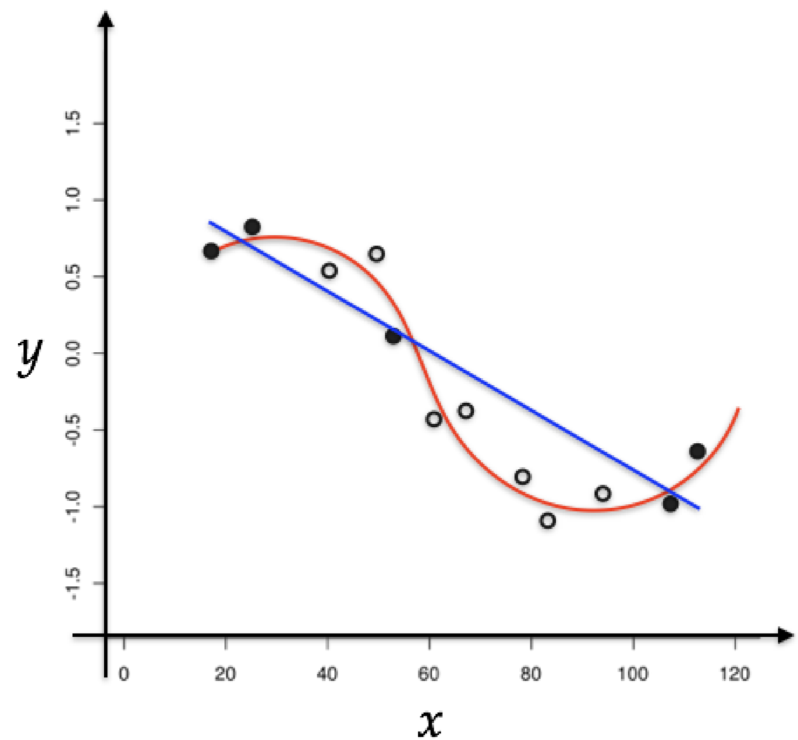
Machine Learning

Machine learning is about extracting {patterns, knowledge, information} from data

Cortona

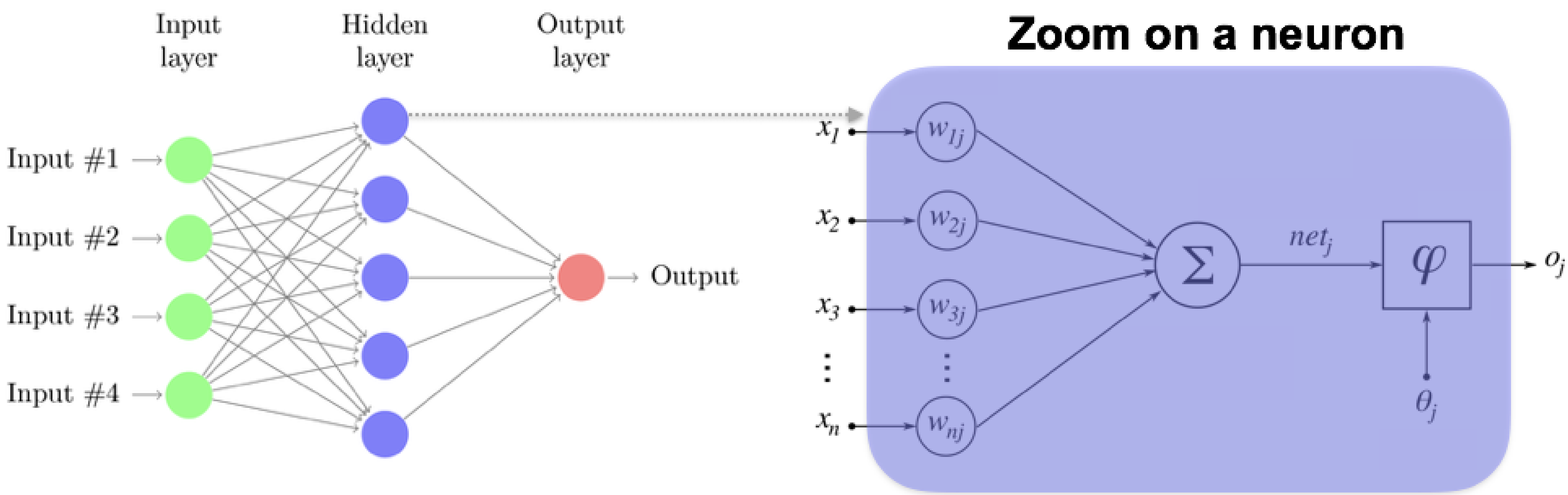
SIRI

OK Google



Deep Learning

Machine learning algorithms have recently shown impressive results, in particular when input data are images: this has led to the identification of a subfield of Machine Learning called Deep Learning.



(Deep) Reinforcement Learning

Reinforcement learning, an area of machine learning originally inspired by behaviorist psychology, concerned with how software agents ought to take actions in an environment so as to maximize some notion of cumulative reward.

Deep reinforcement learning combines deep learning with reinforcement learning (and, consequently, in DP / MPC schemes).

Recent (Deep) Reinforcement Learning Successes

Human-level control through deep reinforcement learning. Nature, 2015.
Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg & Demis Hassabis

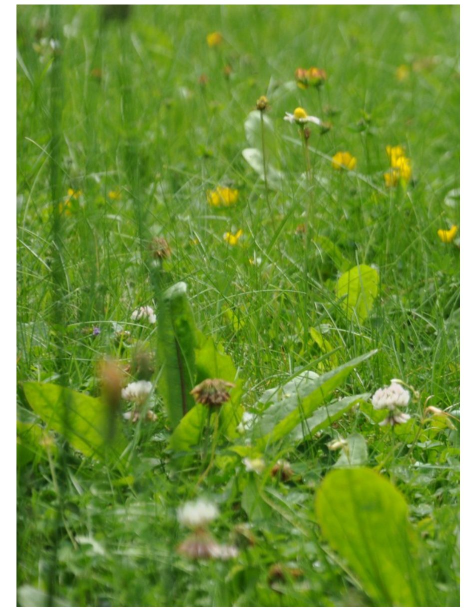
Mastering the game of Go with deep neural networks and tree search. Nature, 2016.
David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel & Demis Hassabis

Batch Mode Reinforcement Learning

Reinforcement Learning

Agent

Environment



Actions

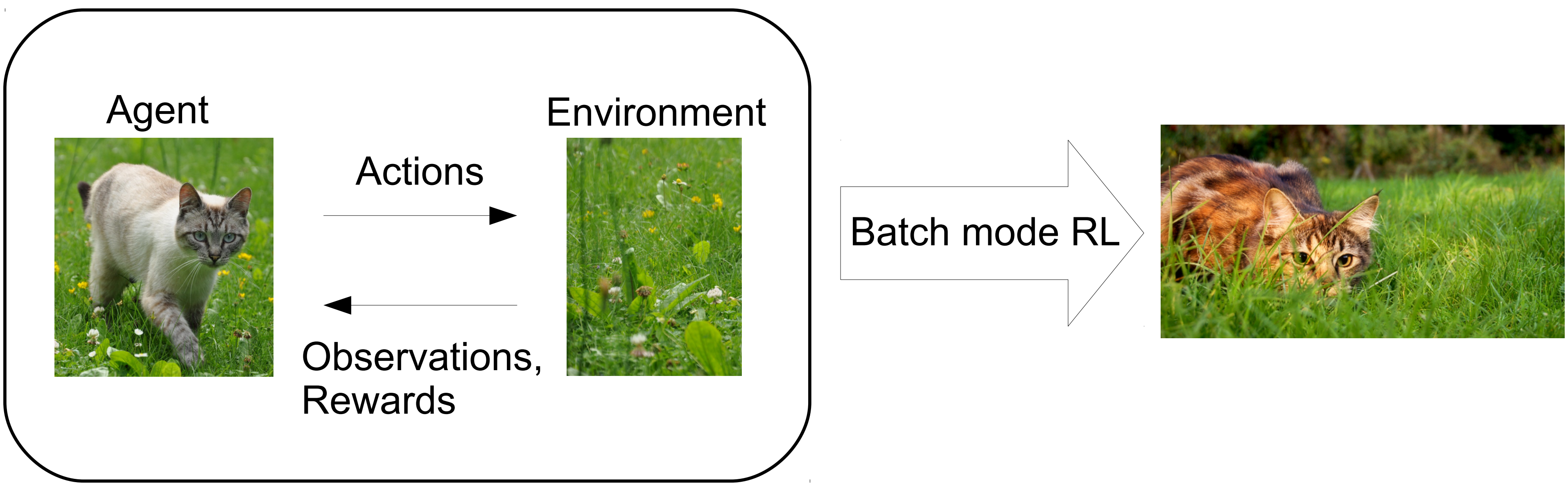
Observations, Rewards

Examples of rewards:

- Reinforcement Learning (RL) aims at **finding a policy maximizing received rewards** by **interacting** with the environment

Batch Mode Reinforcement Learning

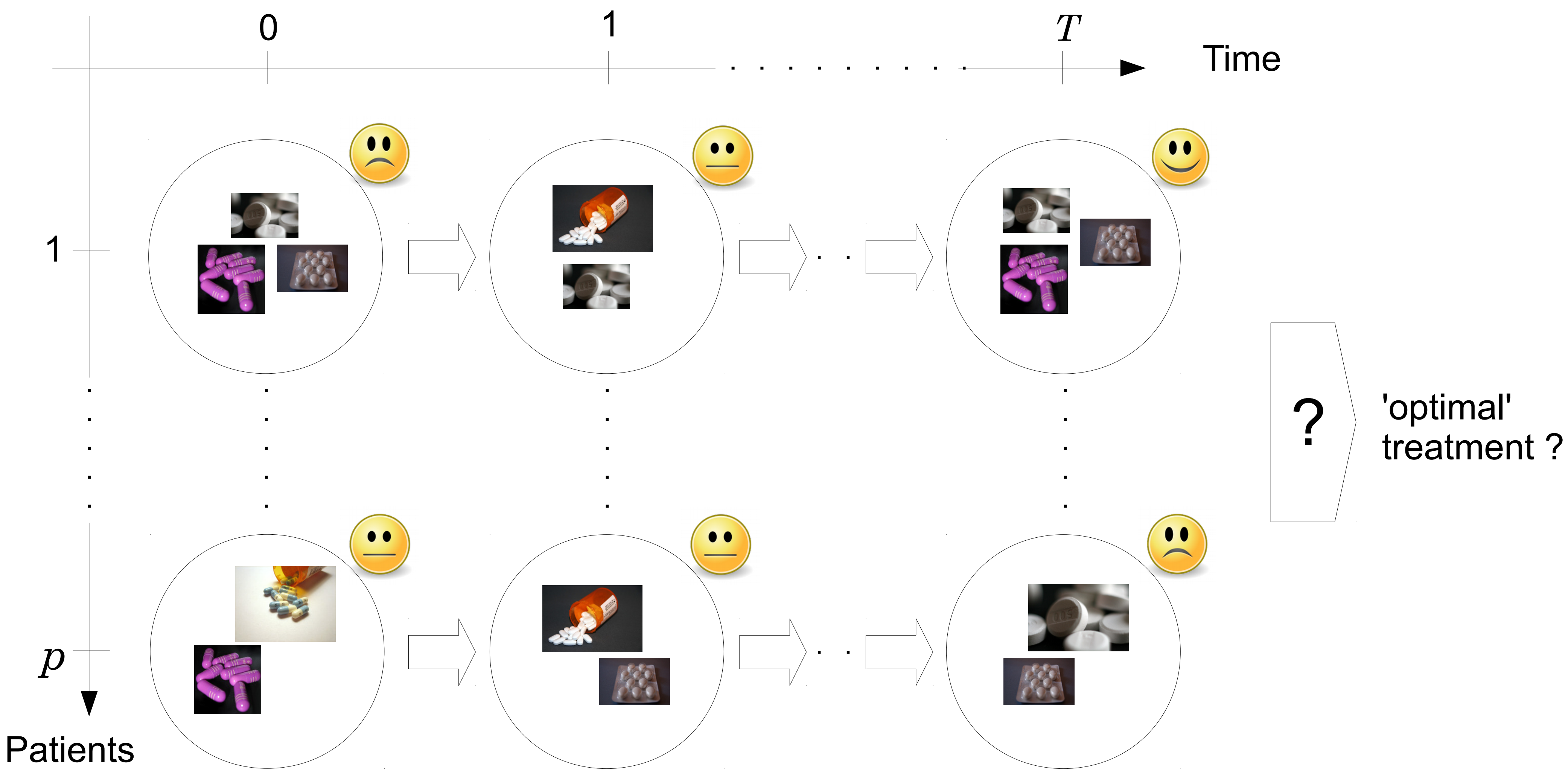
- All the available information is contained in a **batch collection of data**
- Batch mode RL aims at computing a (near-)optimal policy from this collection of data



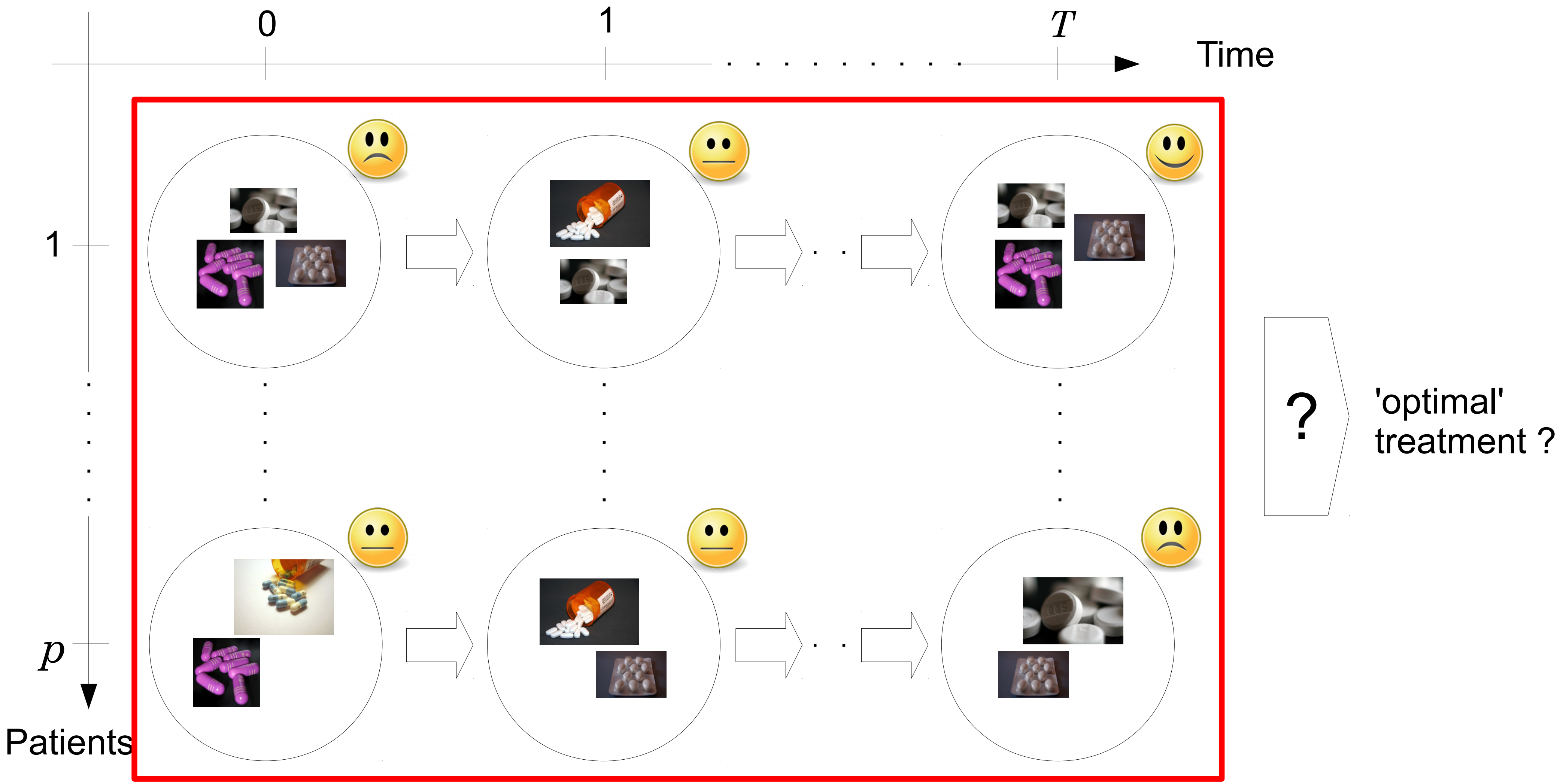
Finite collection of trajectories of the agent

Near-optimal decision strategy

Batch Mode Reinforcement Learning



Batch Mode Reinforcement Learning



Batch collection of trajectories of patients

Objectives

- Main goal: **Finding a "good" policy**

- Many associated subgoals:
  - Evaluating the performance of a given policy
  - Computing performance guarantees
  - Computing safe policies
  - Choosing how to generate additional transitions
  - ...

Main Difficulties

Main difficulties of the batch mode setting:

- Dynamics and reward functions are **unknown** (and not accessible to simulation)
- The state-space and/or the action space are large or **continuous**
- The environment may be highly **stochastic**

Usual Approach

To **combine dynamic programming with function approximators** (neural networks, regression trees, SVM, linear regression over basis functions, etc)

Function approximators have two main roles:

- To offer a **concise representation** of state-action value function for deriving value / policy iteration algorithms
- To **generalize information** contained in the finite sample

Remaining Challenges

The **black box nature of function approximators** may have some unwanted effects:

- hazardous generalization
- difficulties to compute performance guarantees
- unefficient use of optimal trajectories

A proposition: synthesizing artificial trajectories

Synthesizing Artificial Trajectories

Formalization

Reinforcement learning

System dynamics: $x_{t+1} = f\left(x_t, u_t, w_t\right)$

$t \in \{0, \ldots, T-1\}$ $x_t \in \mathcal{X} \subset \mathbb{R}^d$ $u_t \in \mathcal{U}$ $w_t \in \mathcal{W}$

$w_t \sim p_{\mathcal{W}}(\cdot)$

Reward function: $r_t = \rho\left(x_t, u_t, w_t\right)$

Performance of a policy $h : \{0, \ldots, T-1\} \times \mathcal{X} \rightarrow \mathcal{U}$

$$J^h(x_0) = \mathbb{E}\left[R^h(x_0, w_0, \ldots, w_{T-1})\right]$$

$$R^h(x_0, w_0, \ldots, w_{T-1}) = \sum_{t=0}^{T-1} \rho(x_t, h(t, x_t), w_t)$$

where

$$x_{t+1} = f(x_t, h(t, x_t), w_t)$$

Formalization

Batch mode reinforcement learning

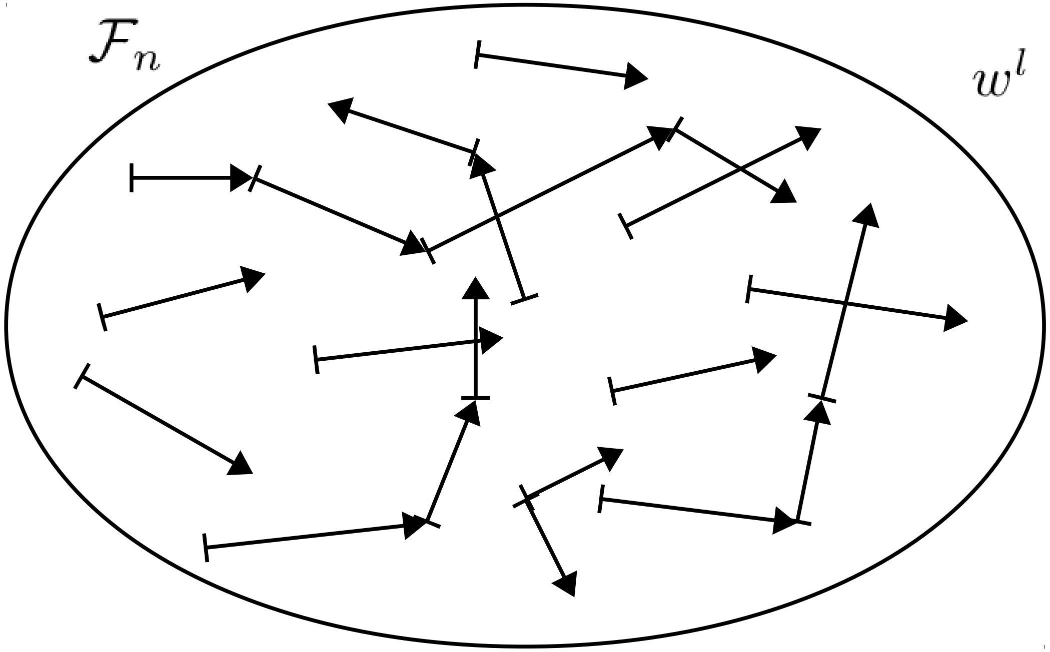
The system dynamics, reward function and disturbance probability distribution are **unknown**

Instead, we have access to a **sample of one-step system transitions**:

$$\mathcal{F}_n = \left\{ \left(x^l, u^l, r^l, y^l \right) \right\}_{l=1}^{n} \qquad \forall l \in \{1, \ldots, n\}, \qquad r^l = \rho\left(x^l, u^l, w^l\right)$$

$$y^l = f\left(x^l, u^l, w^l\right)$$

$$w^l \sim p_{\mathcal{W}}(\cdot)$$



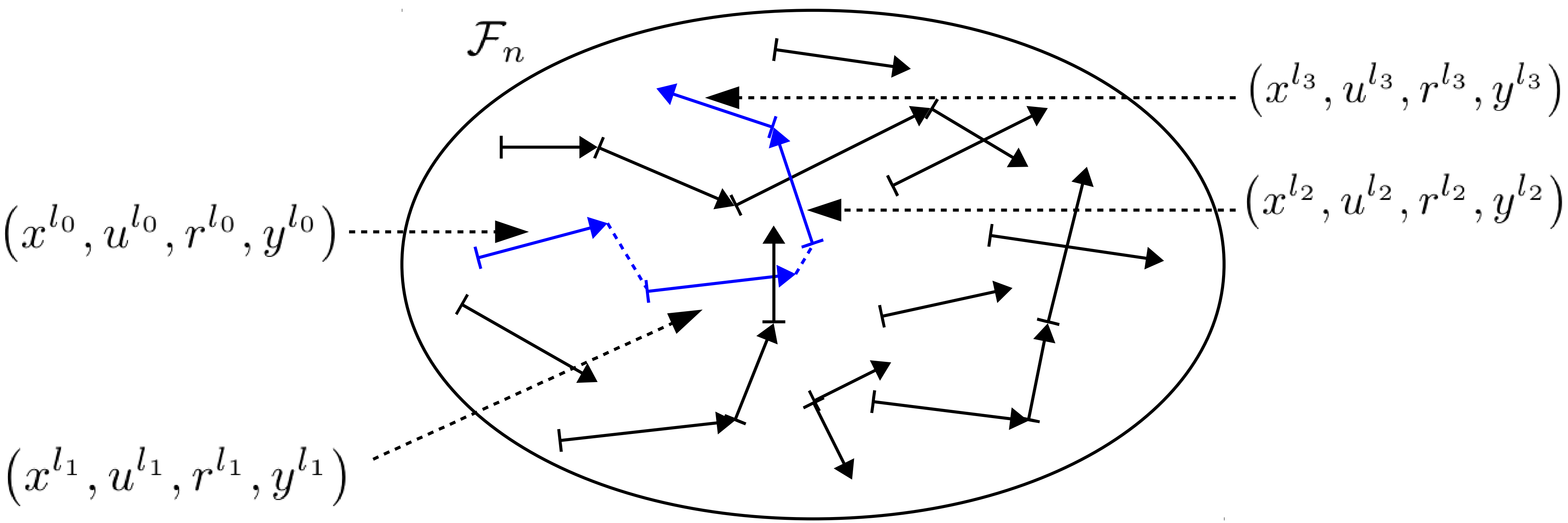
Formalization

Artificial trajectories

Artificial trajectories are **(ordered) sequences of elementary pieces of trajectories:**

$$\left[(x^{l_0}, u^{l_0}, r^{l_0}, y^{l_0}), \ldots, (x^{l_{T-1}}, u^{l_{T-1}}, r^{l_{T-1}}, y^{l_{T-1}}) \right] \in \mathcal{F}_n^T$$

$$l_t \in \{1, \ldots, n\}\ , \qquad \forall t \in \{0, \ldots, T-1\}$$



Artificial Trajectories: What For?

Artificial trajectories can help for:

- Estimating the performances of policies
- Computing performance guarantees
- Computing safe policies
- Choosing how to generate additional transitions

Artificial Trajectories: What For?

Artificial trajectories can help for:

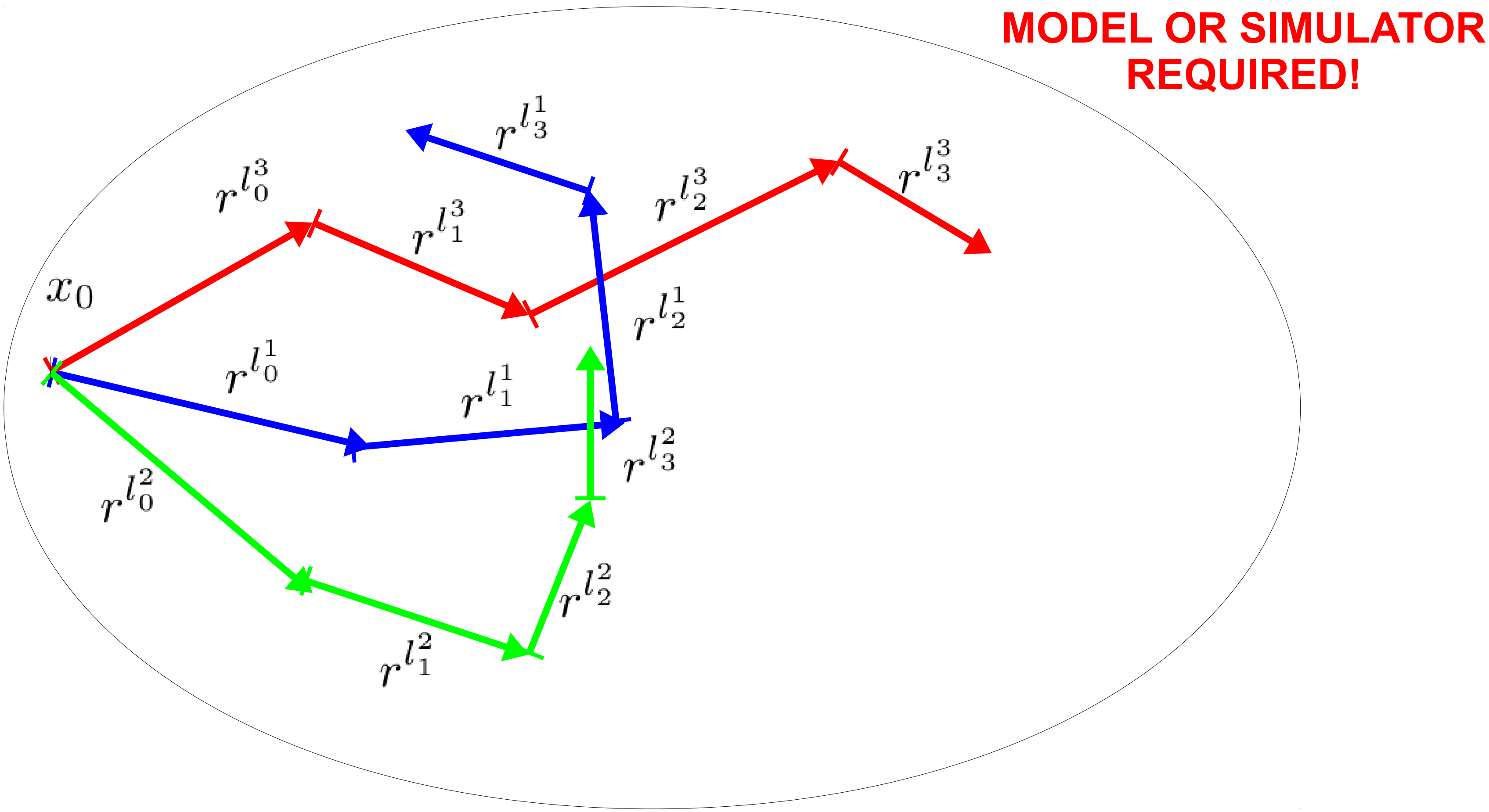
- Estimating the performances of policies
- Computing performance guarantees
- Computing safe policies
- Choosing how to generate additional transitions

Estimating the Performances of Policies

Model-free Monte Carlo Estimation

If the system dynamics and the reward function were accessible to simulation, then **Monte Carlo estimation** would allow estimating the performance of *h*

Model-free Monte Carlo Estimation



$$\mathbb{M}_3^h(x_0) = \frac{\left(r^{l_0^1} + r^{l_1^1} + r^{l_2^1} + r^{l_3^1}\right) + \left(r^{l_0^2} + r^{l_1^2} + r^{l_2^2} + r^{l_3^2}\right) + \left(r^{l_0^3} + r^{l_1^3} + r^{l_2^3} + r^{l_3^3}\right)}{3}$$

Model-free Monte Carlo Estimation

If the system dynamics and the reward function were accessible to simulation, then **Monte Carlo (MC) estimation** would allow estimating the performance of h

We propose an approach that mimics MC estimation by rebuilding p **artificial trajectories** from one-step system transitions

Model-free Monte Carlo Estimation

If the system dynamics and the reward function were accessible to simulation, then **Monte Carlo (MC) estimation** would allow estimating the performance of *h*

We propose an approach that mimics MC estimation by rebuilding *p* **artificial trajectories** from one-step system transitions

These artificial trajectories are built so as to **minimize the discrepancy (using a distance metric Δ) with a classical MC sample** that could be obtained by simulating the system with the policy *h*; each one step transition is used **at most once**

Model-free Monte Carlo Estimation

If the system dynamics and the reward function were accessible to simulation, then **Monte Carlo (MC) estimation** would allow estimating the performance of *h*

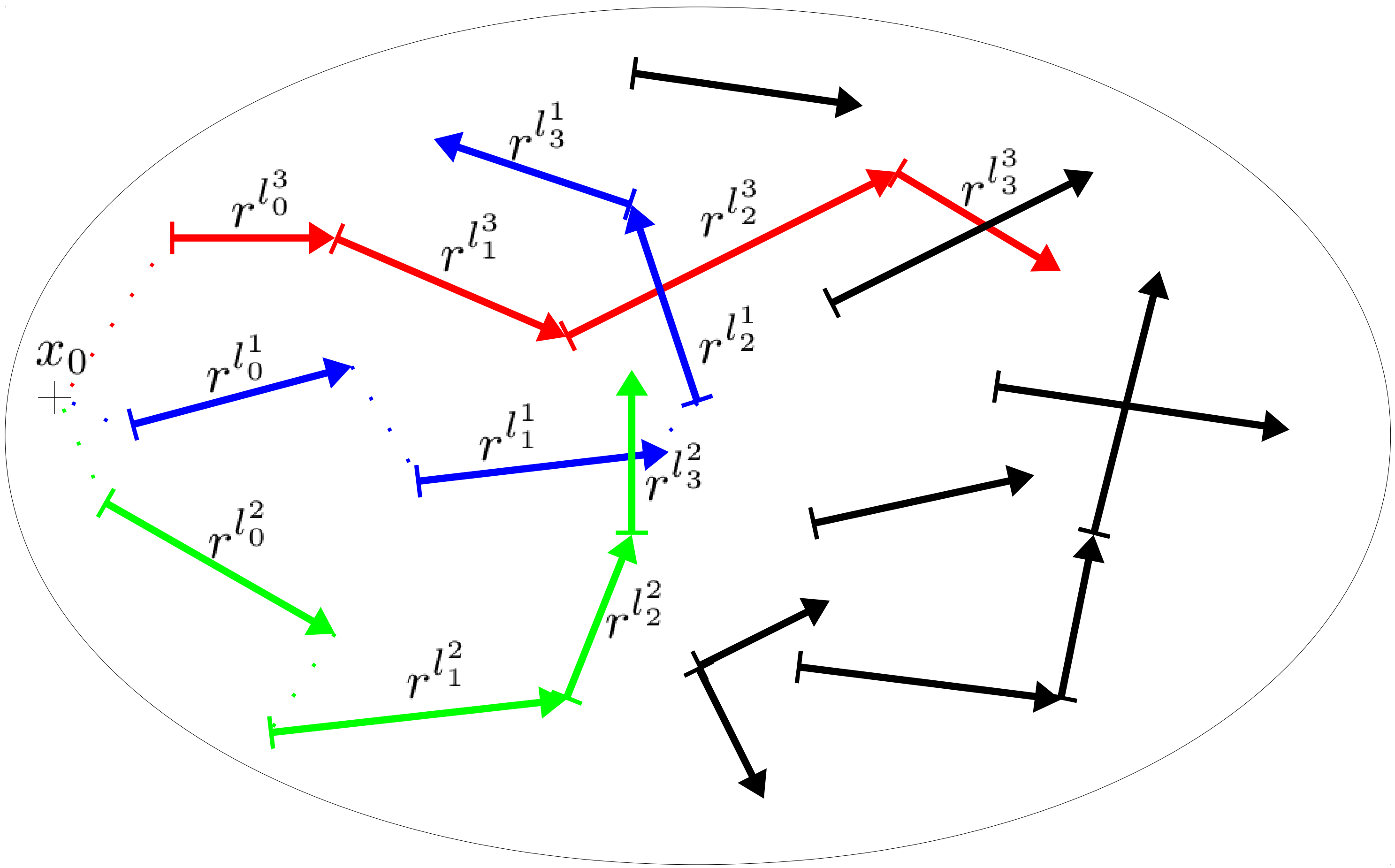
We propose an approach that mimics MC estimation by rebuilding *p* **artificial trajectories** from one-step system transitions

These artificial trajectories are built so as to **minimize the discrepancy (using a distance metric Δ) with a classical MC sample** that could be obtained by simulating the system with the policy *h*; each one step transition is used **at most once**

We average the cumulated returns over the *p* artificial trajectories to obtain the **Model-free Monte Carlo estimator** (MFMC) of the expected return of *h*:

$$\mathfrak{M}_p^h\left(\mathcal{F}_n, x_0\right) = \frac{1}{p}\sum_{i=1}^{p}\sum_{t=0}^{T-1} r^{l_t^i}$$

Model-free Monte Carlo Estimation

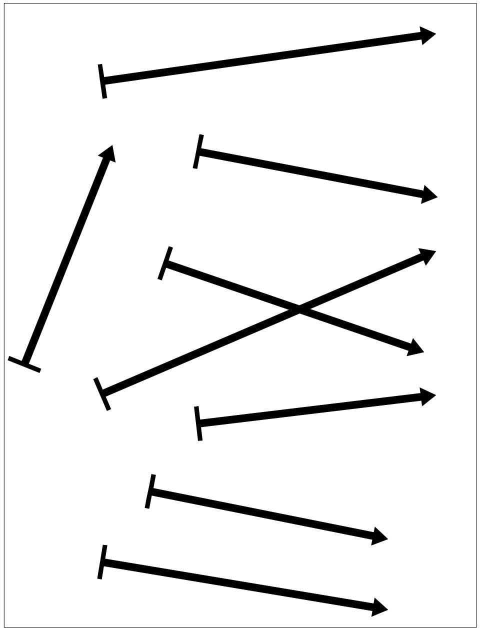


$$\mathfrak{M}_3^h\left(\mathcal{F}_n, x_0\right) = \frac{\left(r^{l_0^1} + r^{l_1^1} + r^{l_2^1} + r^{l_3^1}\right) + \left(r^{l_0^2} + r^{l_1^2} + r^{l_2^2} + r^{l_3^2}\right) + \left(r^{l_0^3} + r^{l_1^3} + r^{l_2^3} + r^{l_3^3}\right)}{3}$$

The MFMC algorithm

Example with T = 3, p = 2, n = 8

$$\mathcal{F}_n = \left\{ \left(x^l, u^l, r^l, y^l \right) \in \mathcal{X} \times \mathcal{U} \times \mathbb{R} \times \mathcal{X} \right\}_{l=1}^{n}$$



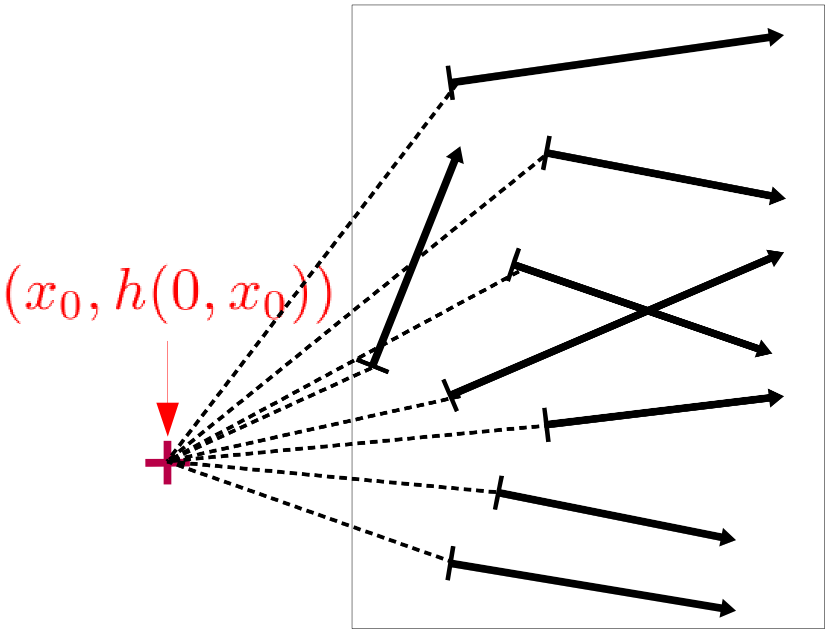
The MFMC algorithm

$(x_0, h(0, x_0))$

$$\downarrow$$

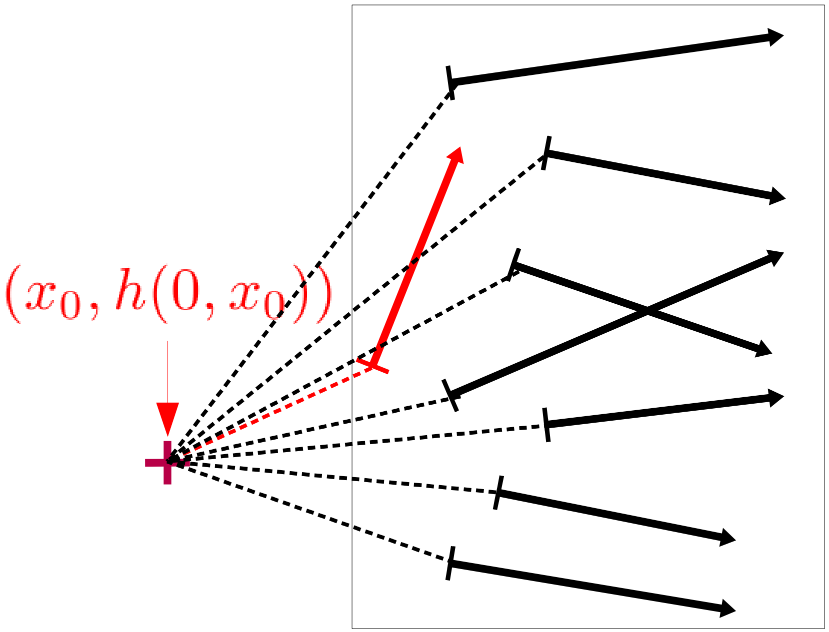
$+$

The MFMC algorithm



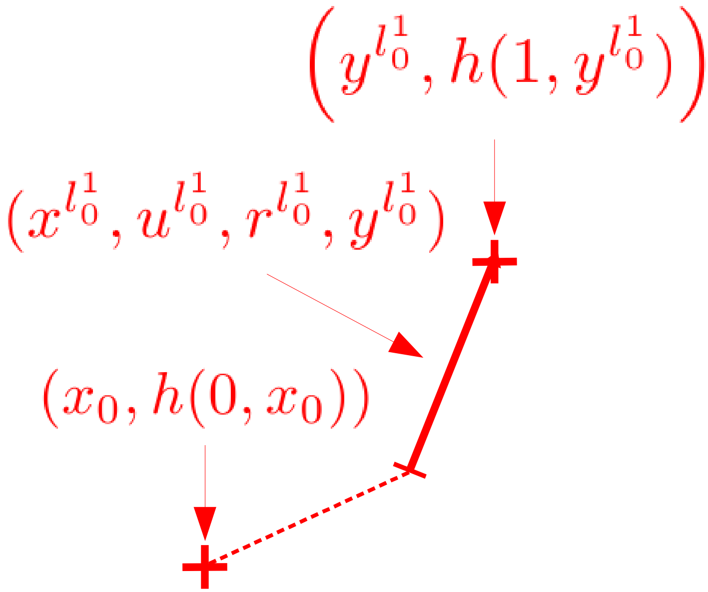
$\mathcal{G} = \mathcal{F}_n$

The MFMC algorithm

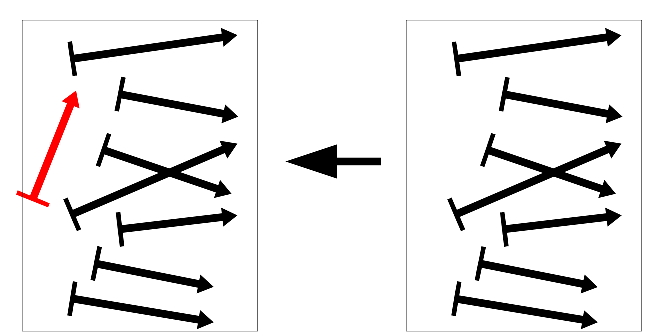


$$\mathcal{G} = \mathcal{F}_n$$

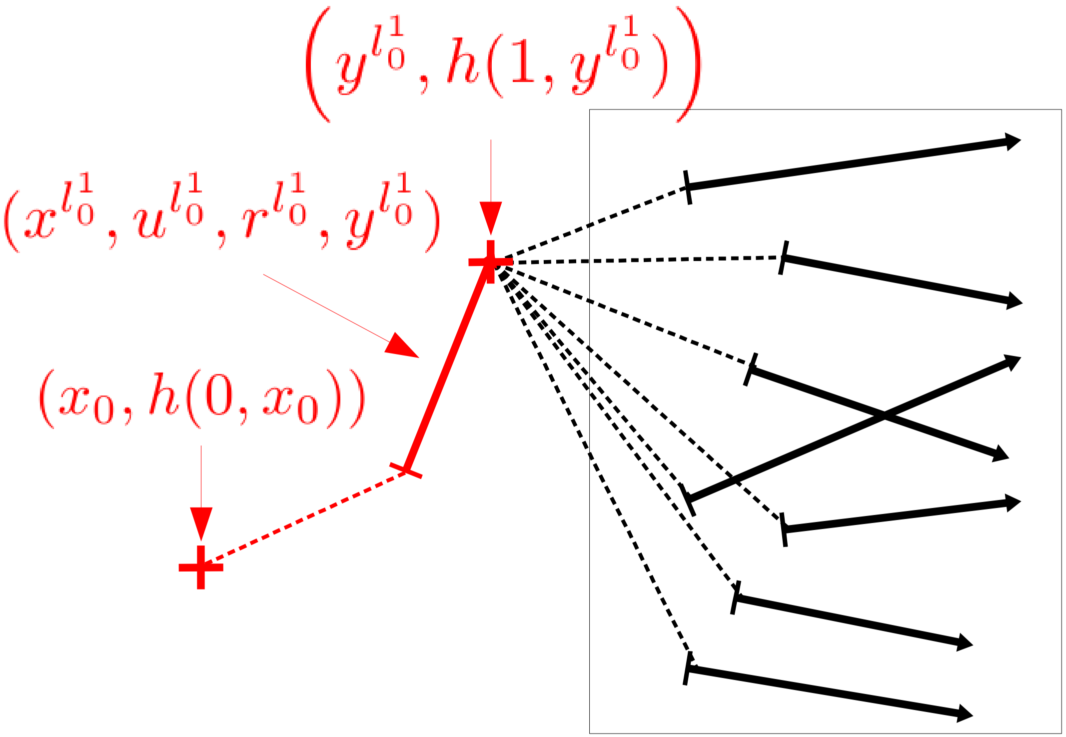
The MFMC algorithm



$$\mathcal{G} = \mathcal{G} \setminus \{(x^{l_0^1}, u^{l_0^1}, r^{l_0^1}, y^{l_0^1})\}$$

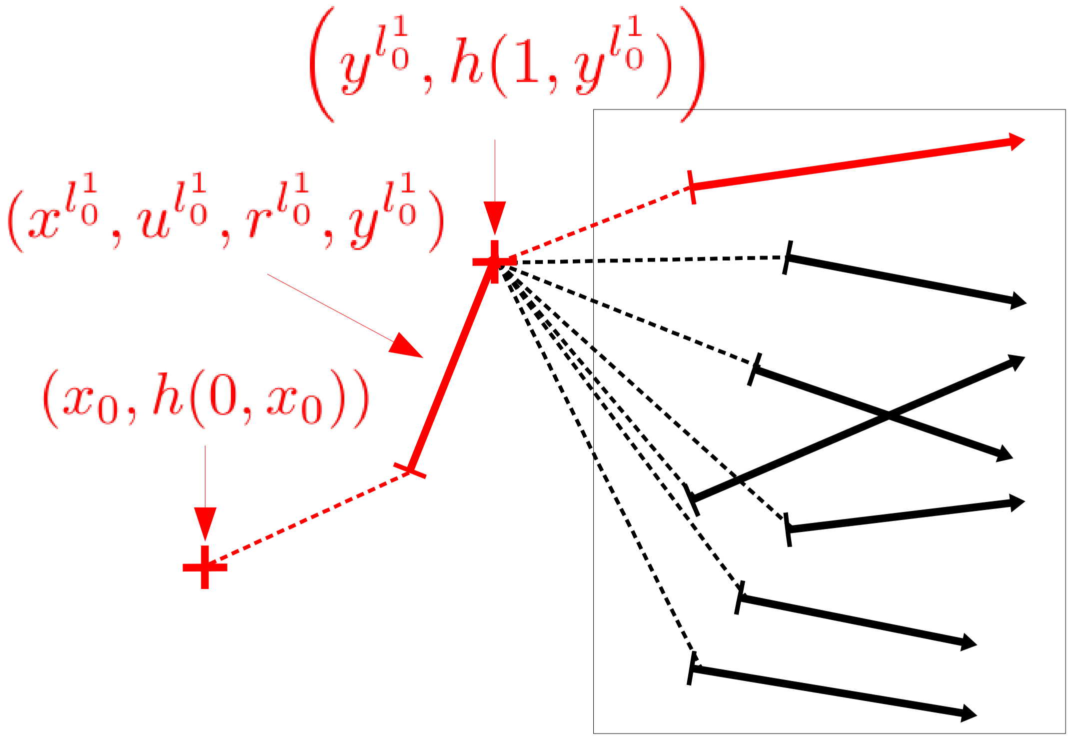


The MFMC algorithm



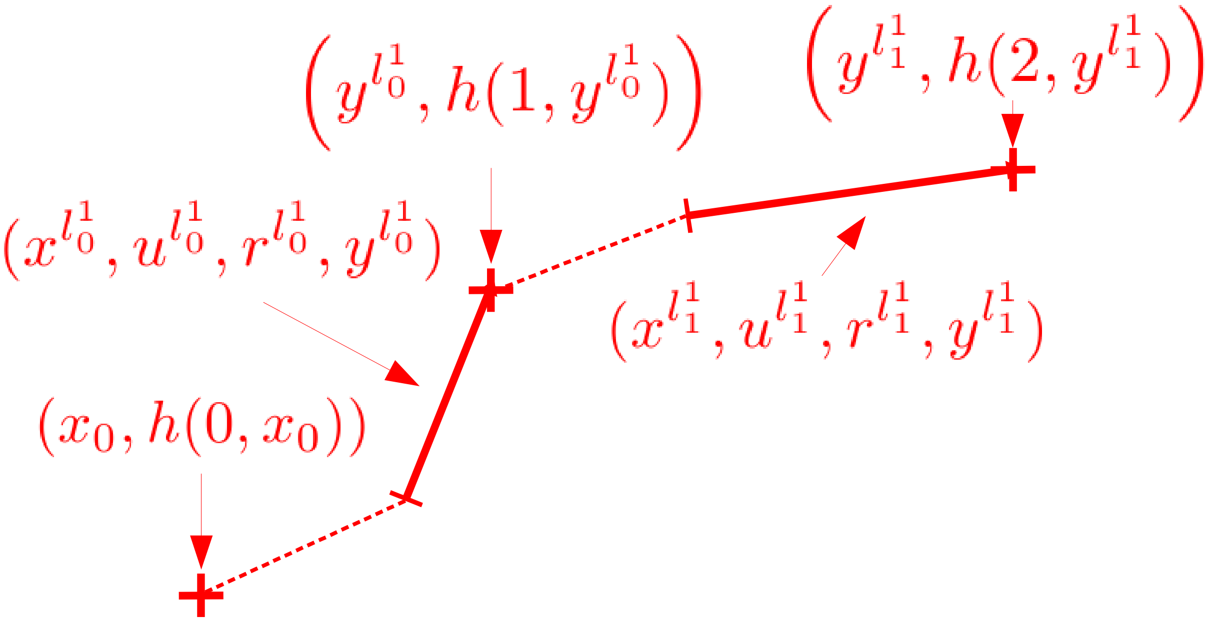
$$\mathcal{G} = \mathcal{G} \setminus \{(x^{l_0^1}, u^{l_0^1}, r^{l_0^1}, y^{l_0^1})\}$$

The MFMC algorithm

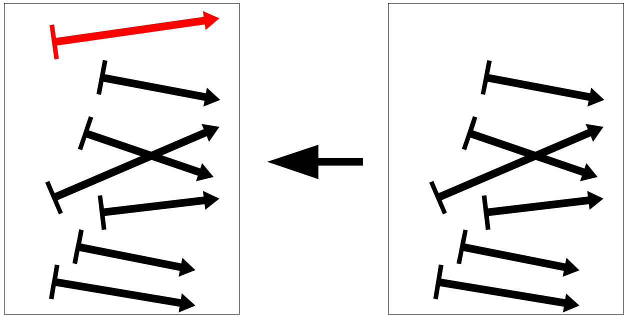


$$\mathcal{G} = \mathcal{G} \setminus \{(x^{l_0^1}, u^{l_0^1}, r^{l_0^1}, y^{l_0^1})\}$$

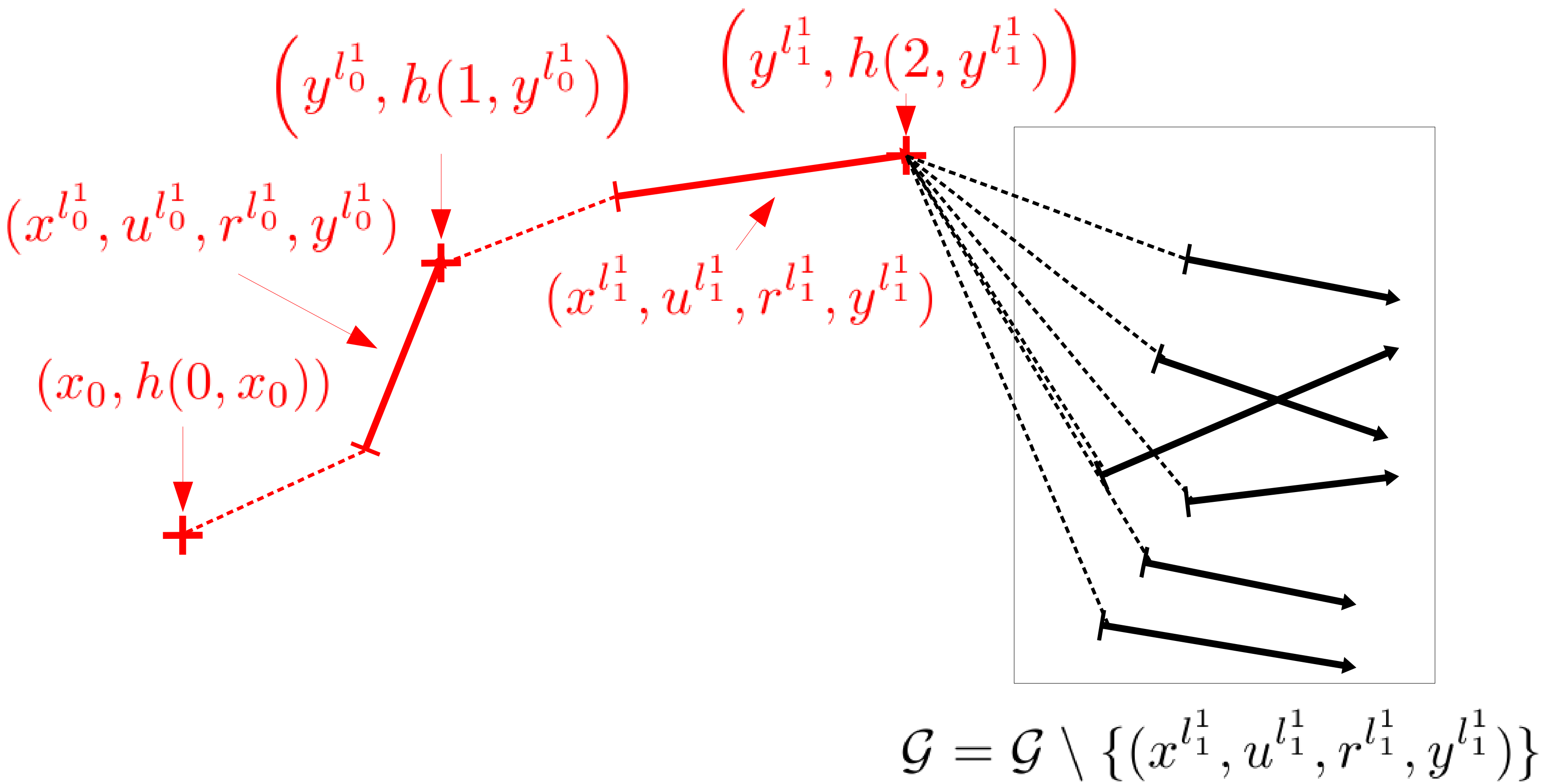
The MFMC algorithm



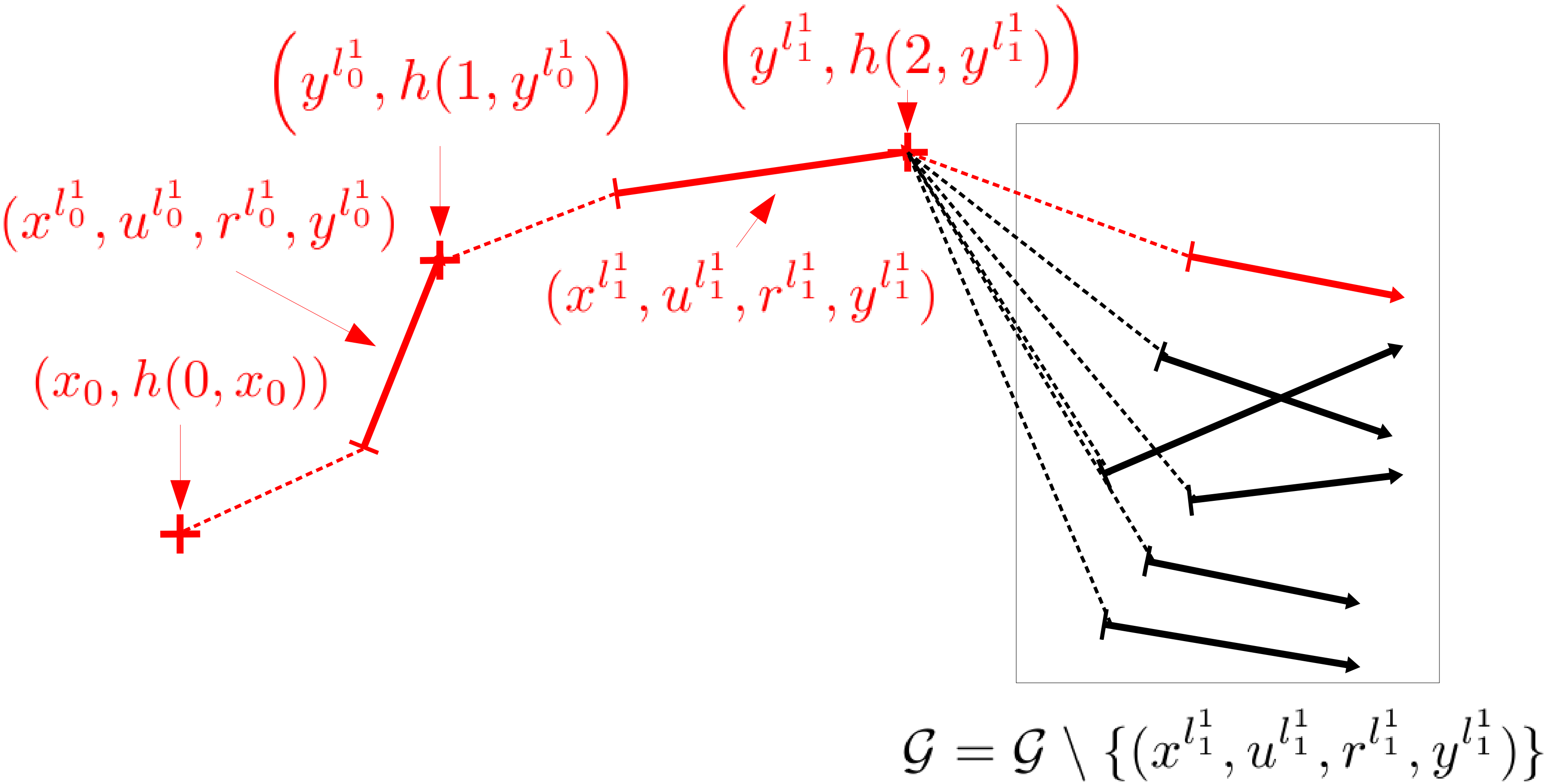
$$\mathcal{G} = \mathcal{G} \setminus \{(x^{l_1^1}, u^{l_1^1}, r^{l_1^1}, y^{l_1^1})\}$$



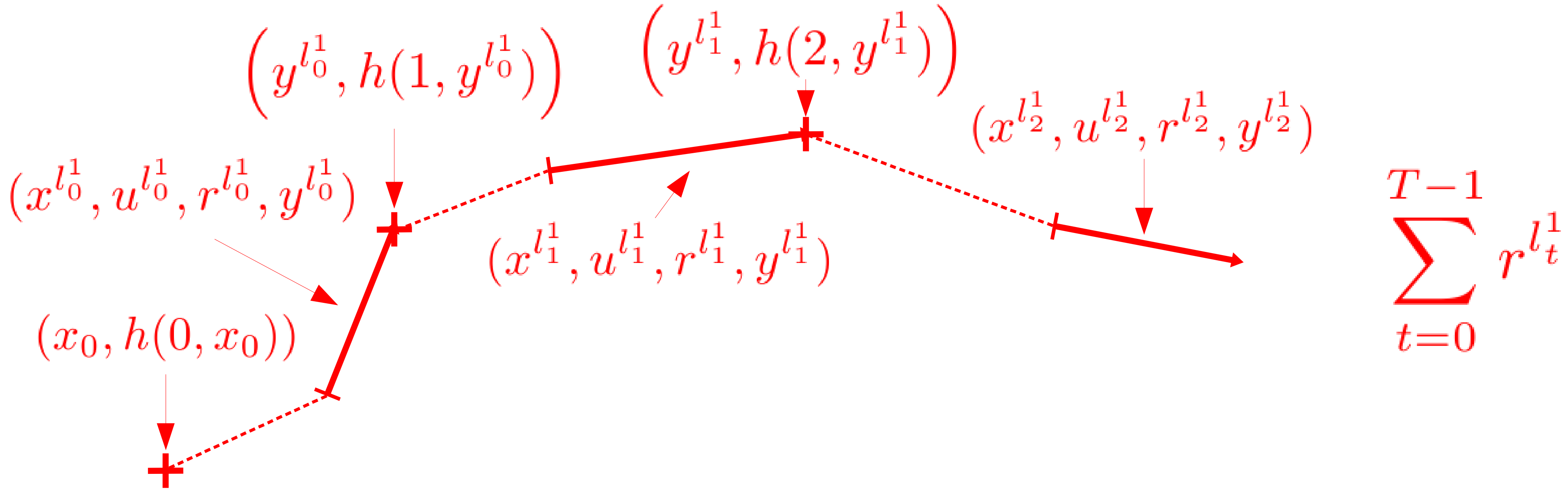
The MFMC algorithm



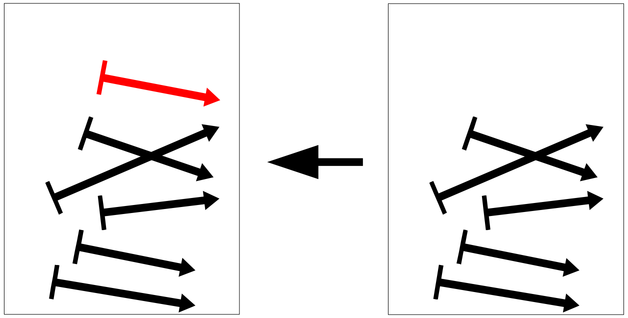
The MFMC algorithm



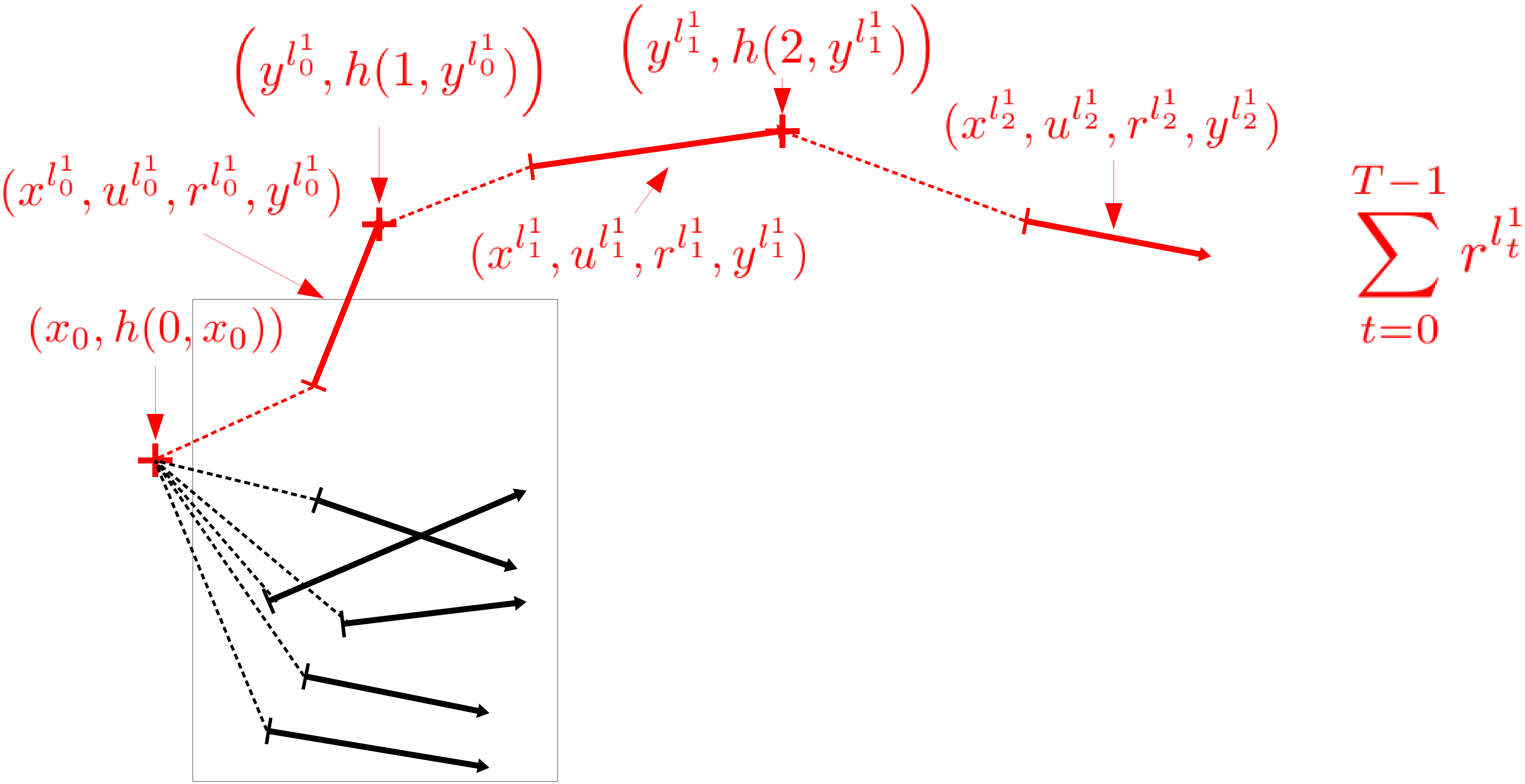
The MFMC algorithm



$$\mathcal{G} = \mathcal{G} \setminus \{(x^{l_2^1}, u^{l_2^1}, r^{l_2^1}, y^{l_2^1})\}$$

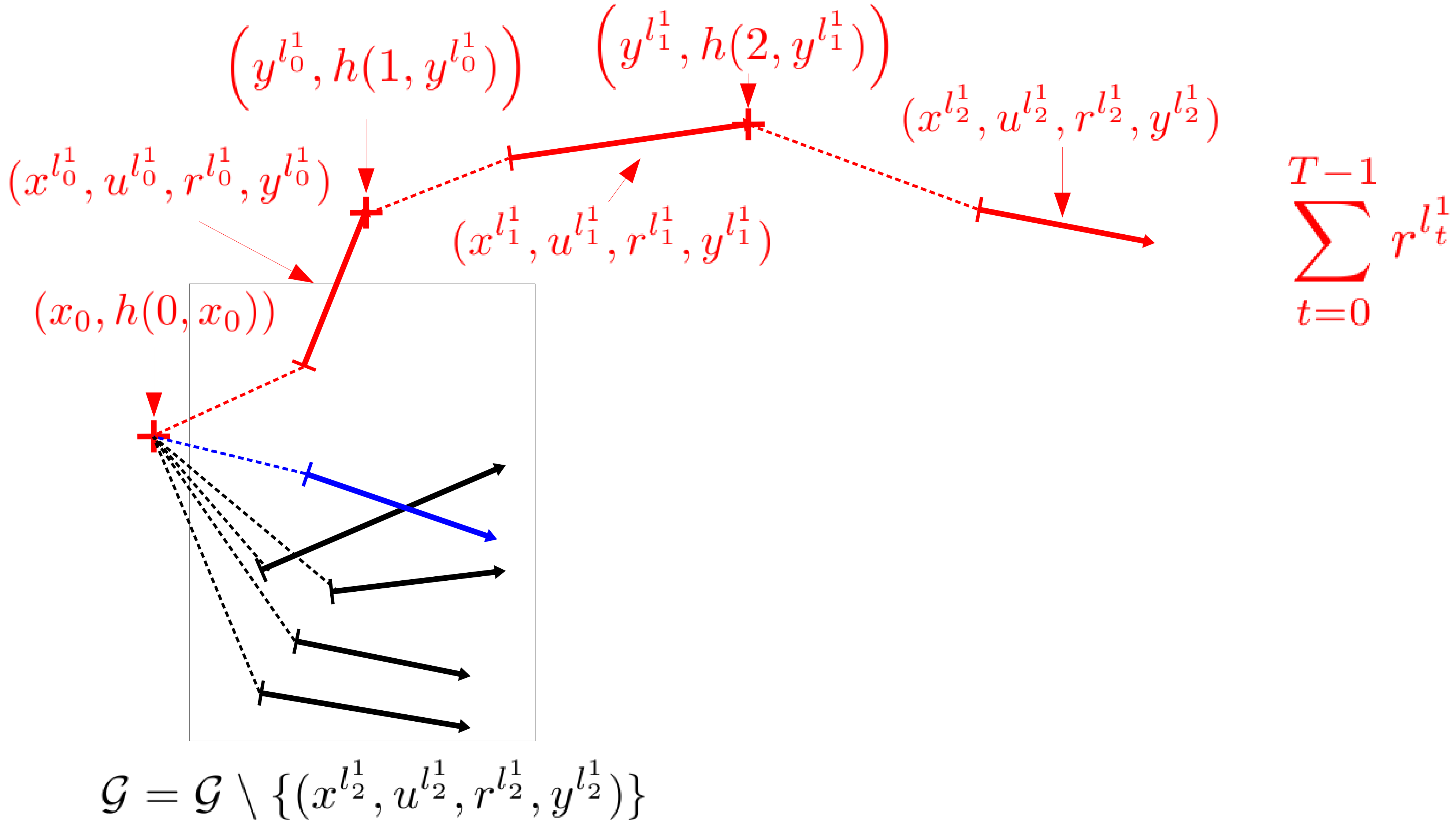


The MFMC algorithm

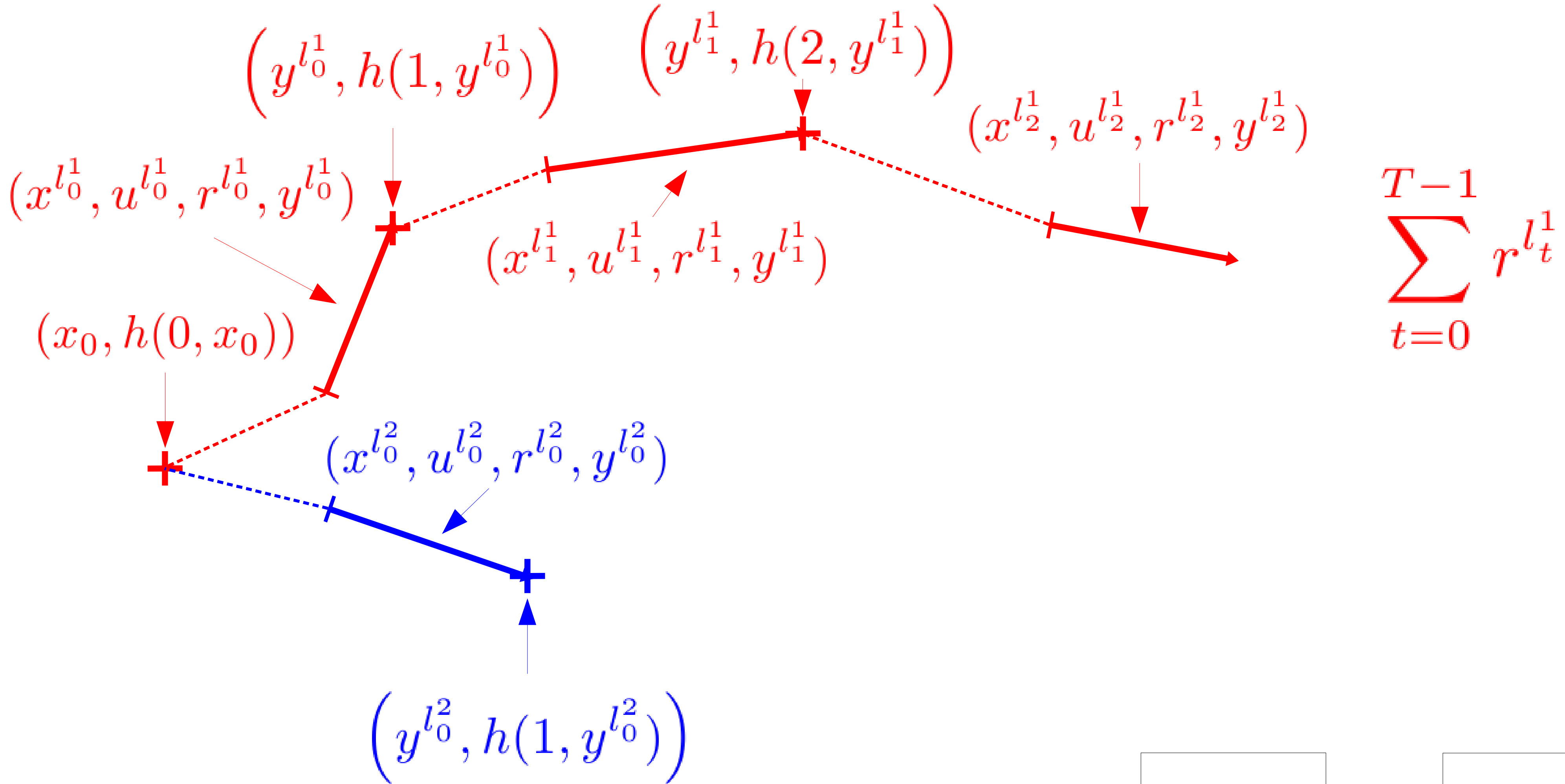


$$\mathcal{G} = \mathcal{G} \setminus \{(x^{l_2^1}, u^{l_2^1}, r^{l_2^1}, y^{l_2^1})\}$$

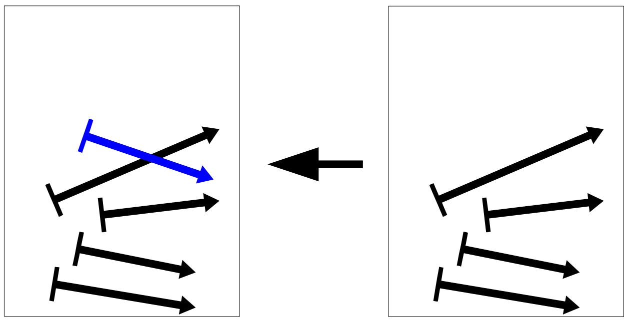
The MFMC algorithm



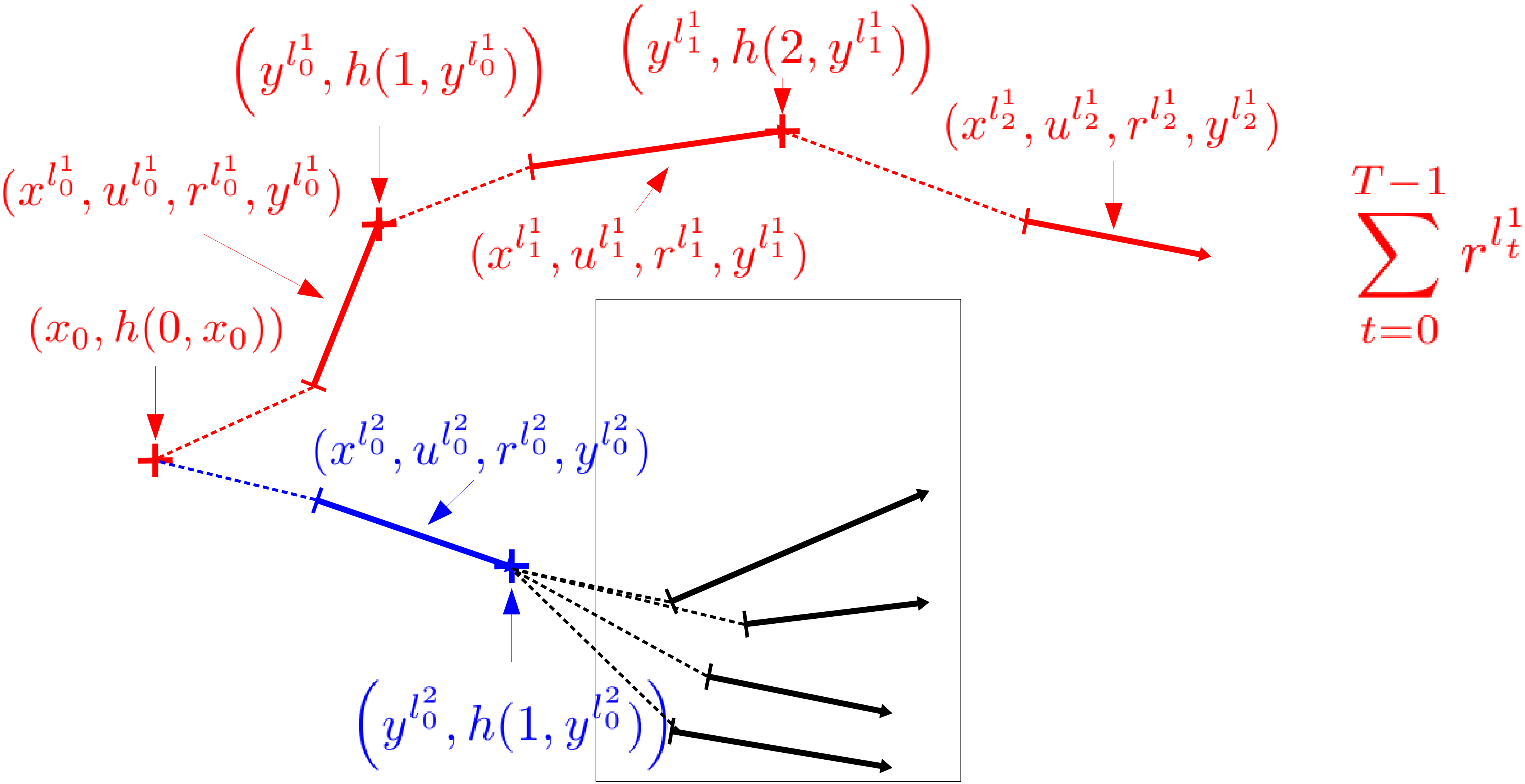
The MFMC algorithm



$$\mathcal{G} = \mathcal{G} \setminus \{(x^{l_0^2}, u^{l_0^2}, r^{l_0^2}, y^{l_0^2})\}$$

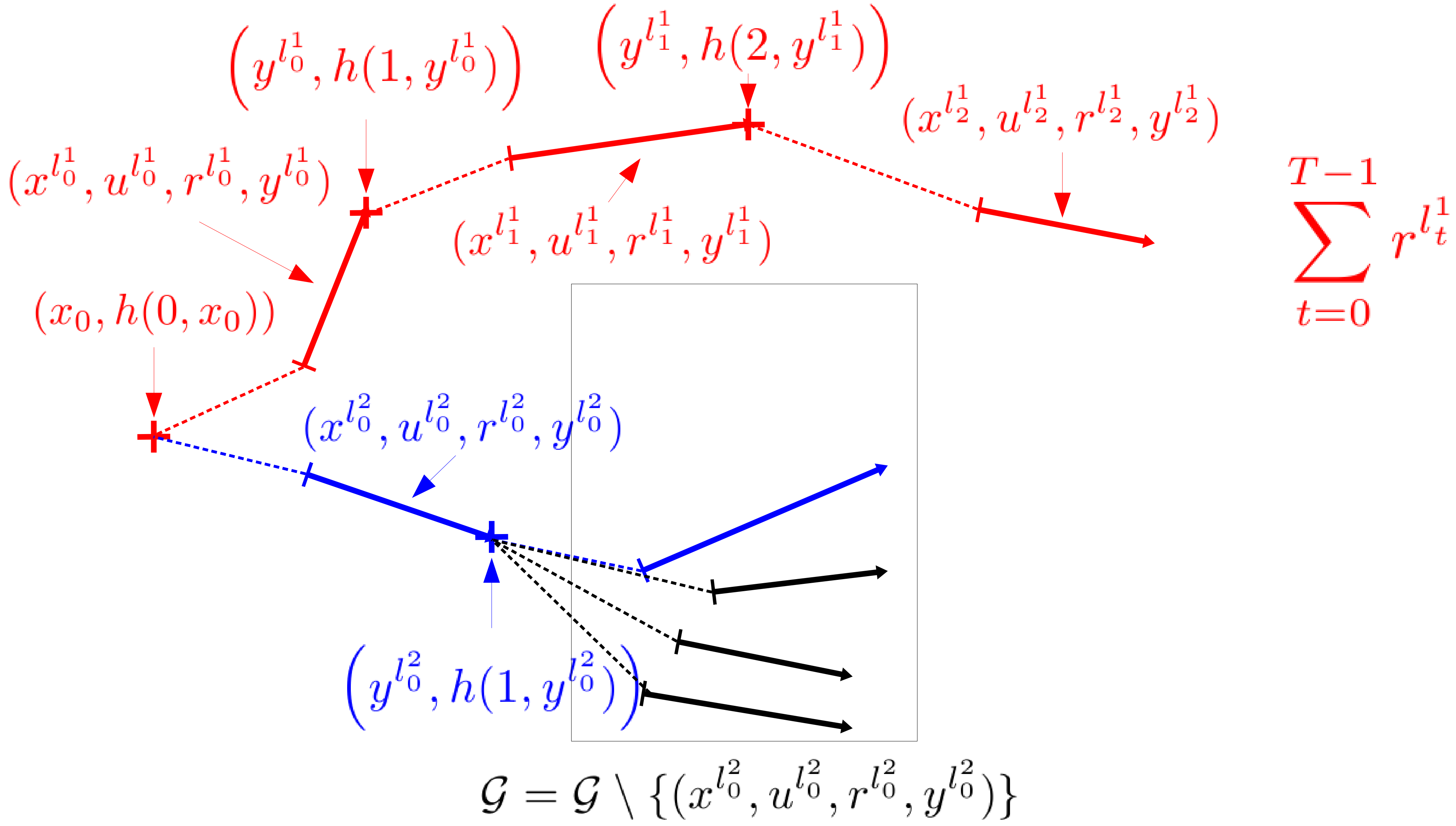


The MFMC algorithm

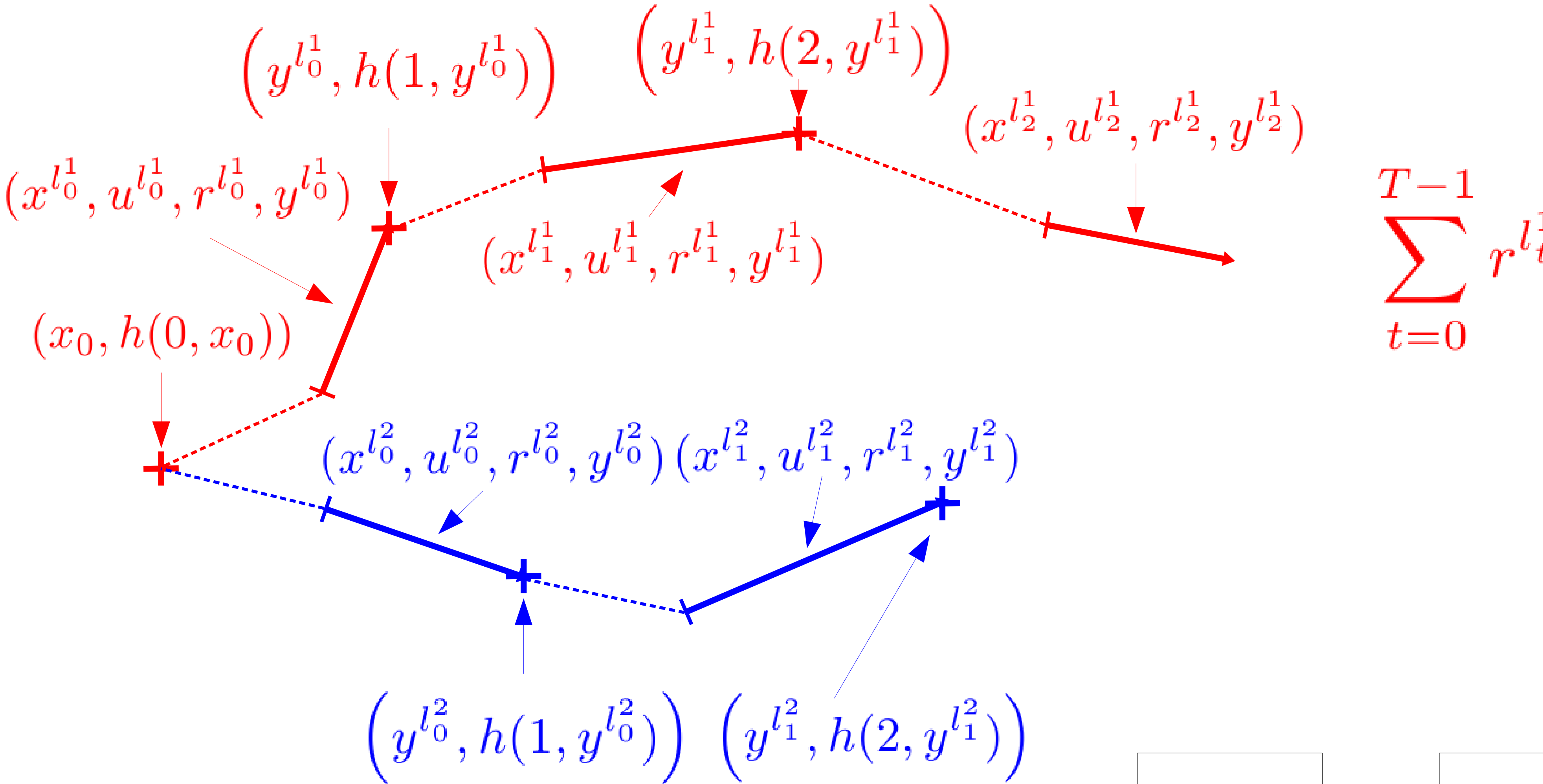


$$\mathcal{G} = \mathcal{G} \setminus \{(x^{l_0^2}, u^{l_0^2}, r^{l_0^2}, y^{l_0^2})\}$$

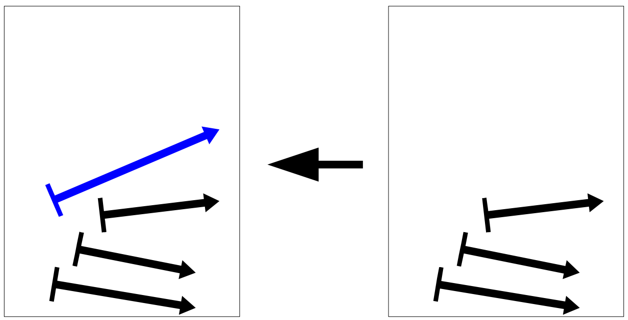
The MFMC algorithm



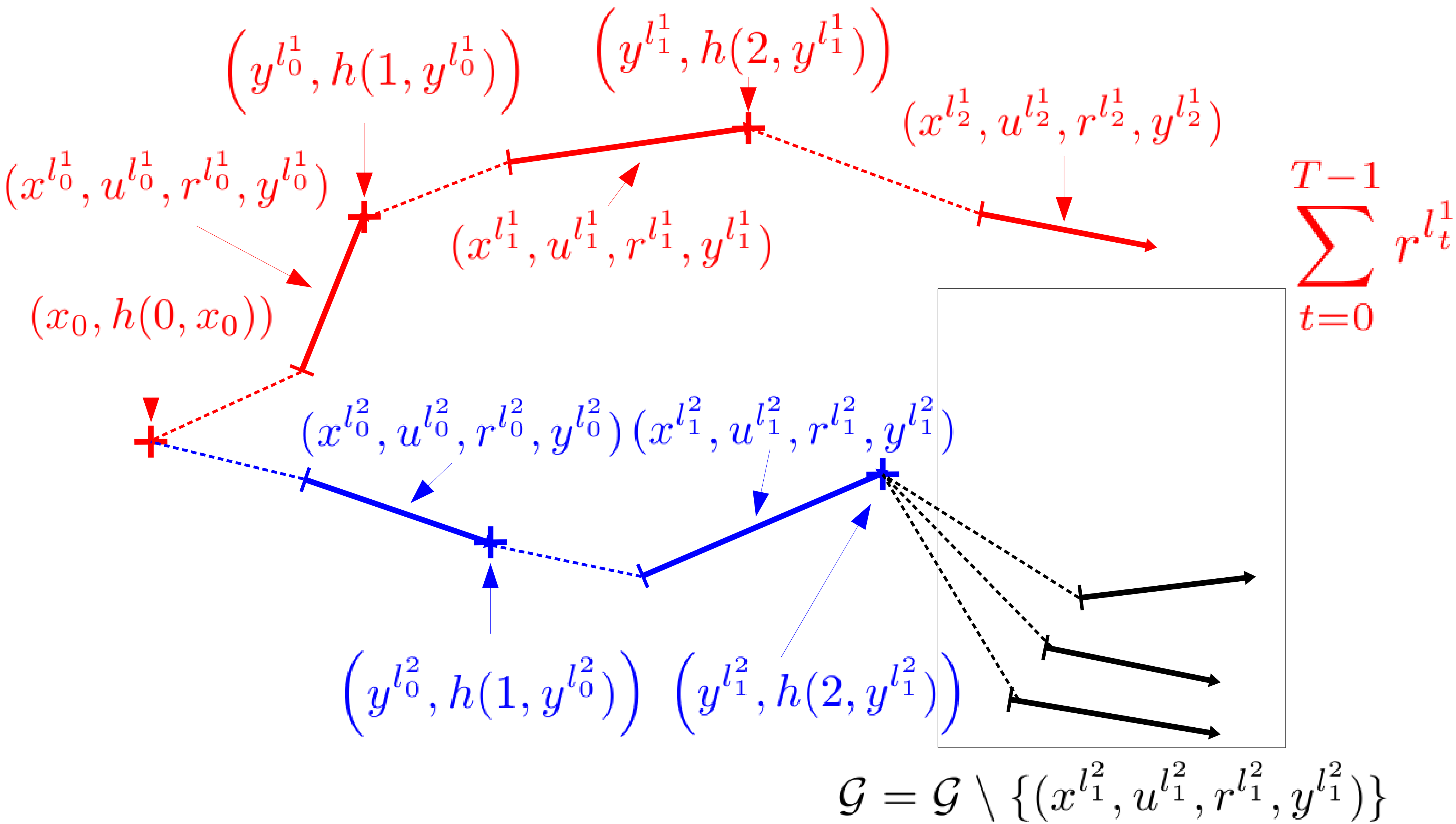
The MFMC algorithm



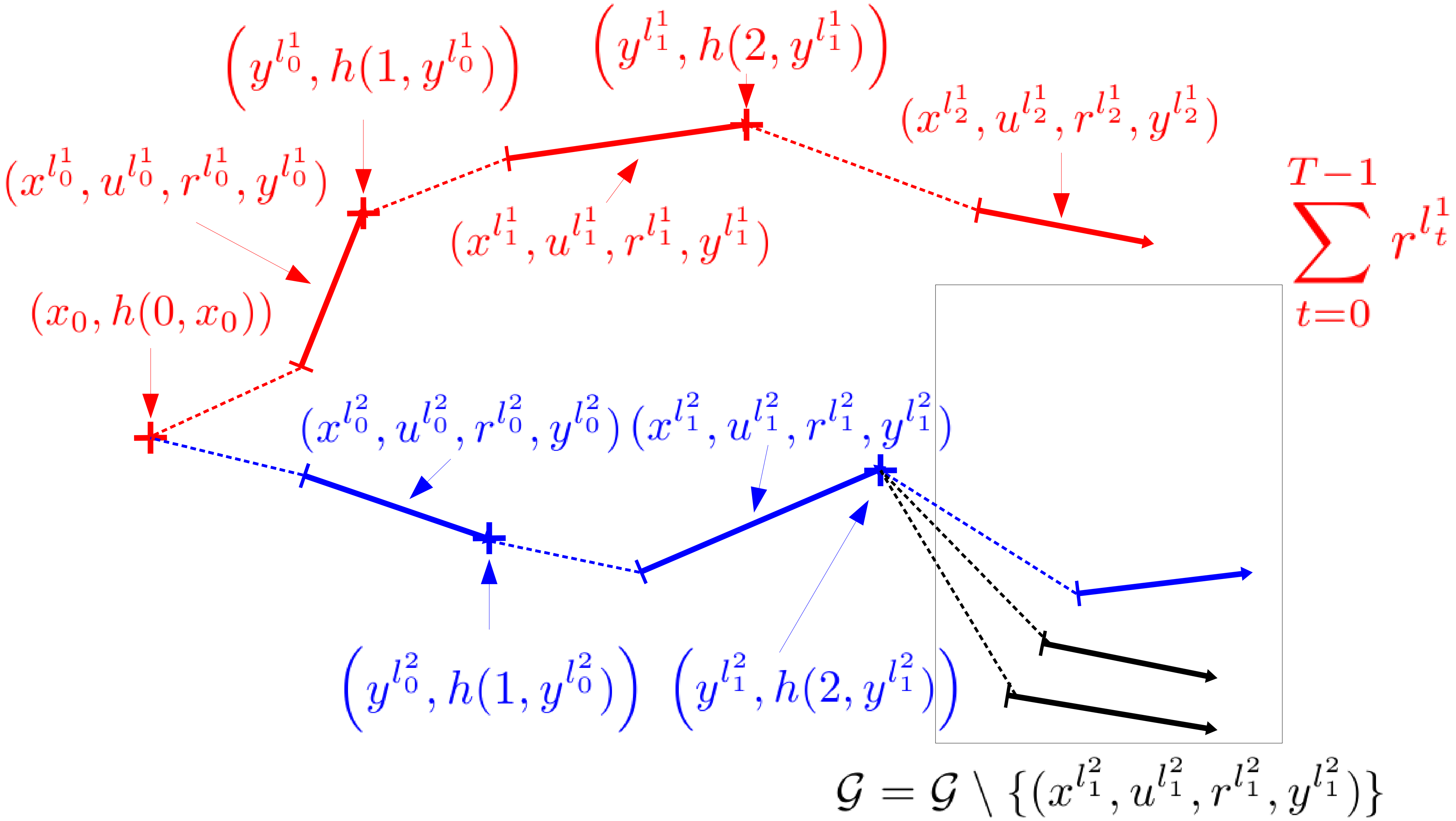
$$\mathcal{G} = \mathcal{G} \setminus \{(x^{l_1^2}, u^{l_1^2}, r^{l_1^2}, y^{l_1^2})\}$$



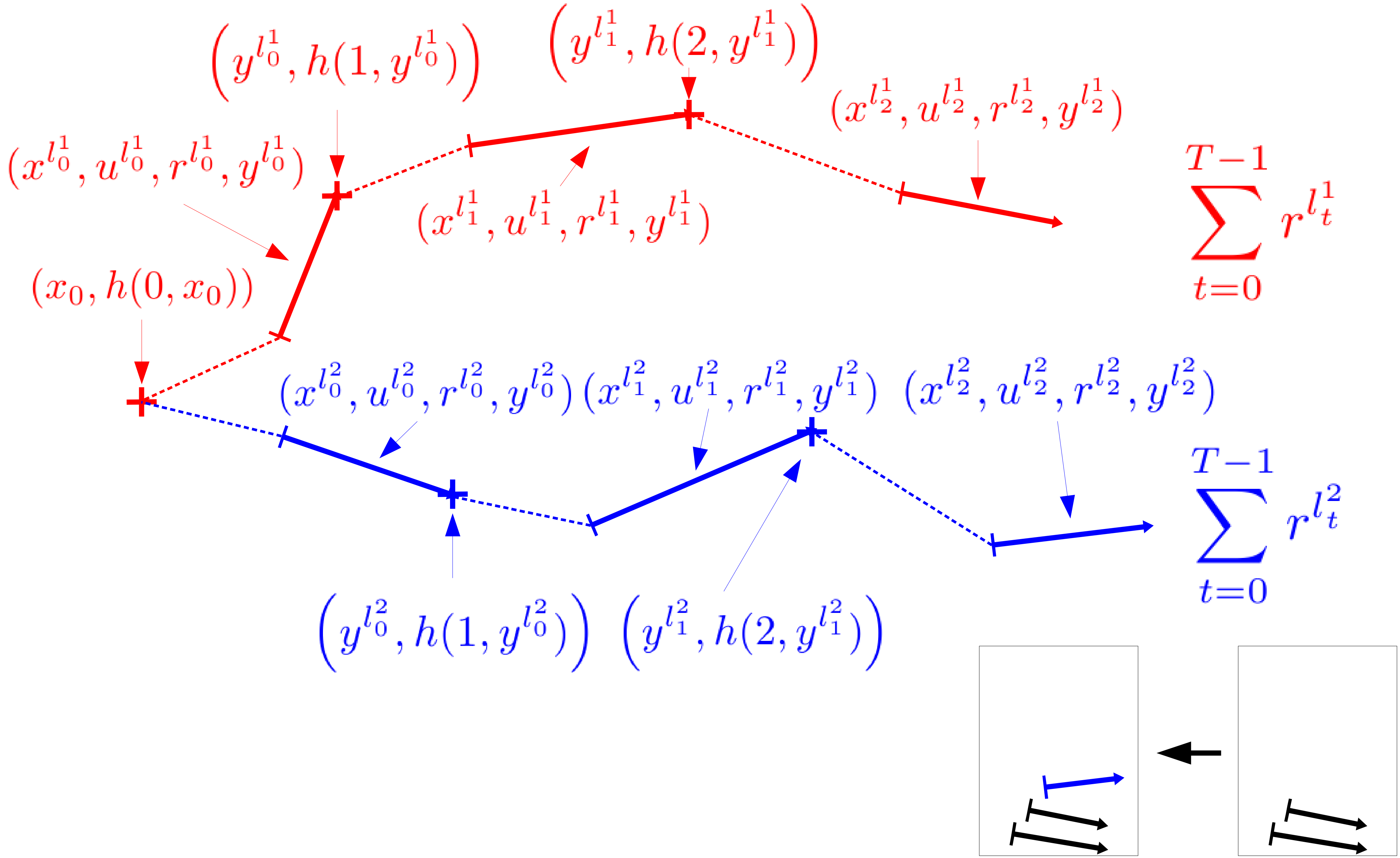
The MFMC algorithm



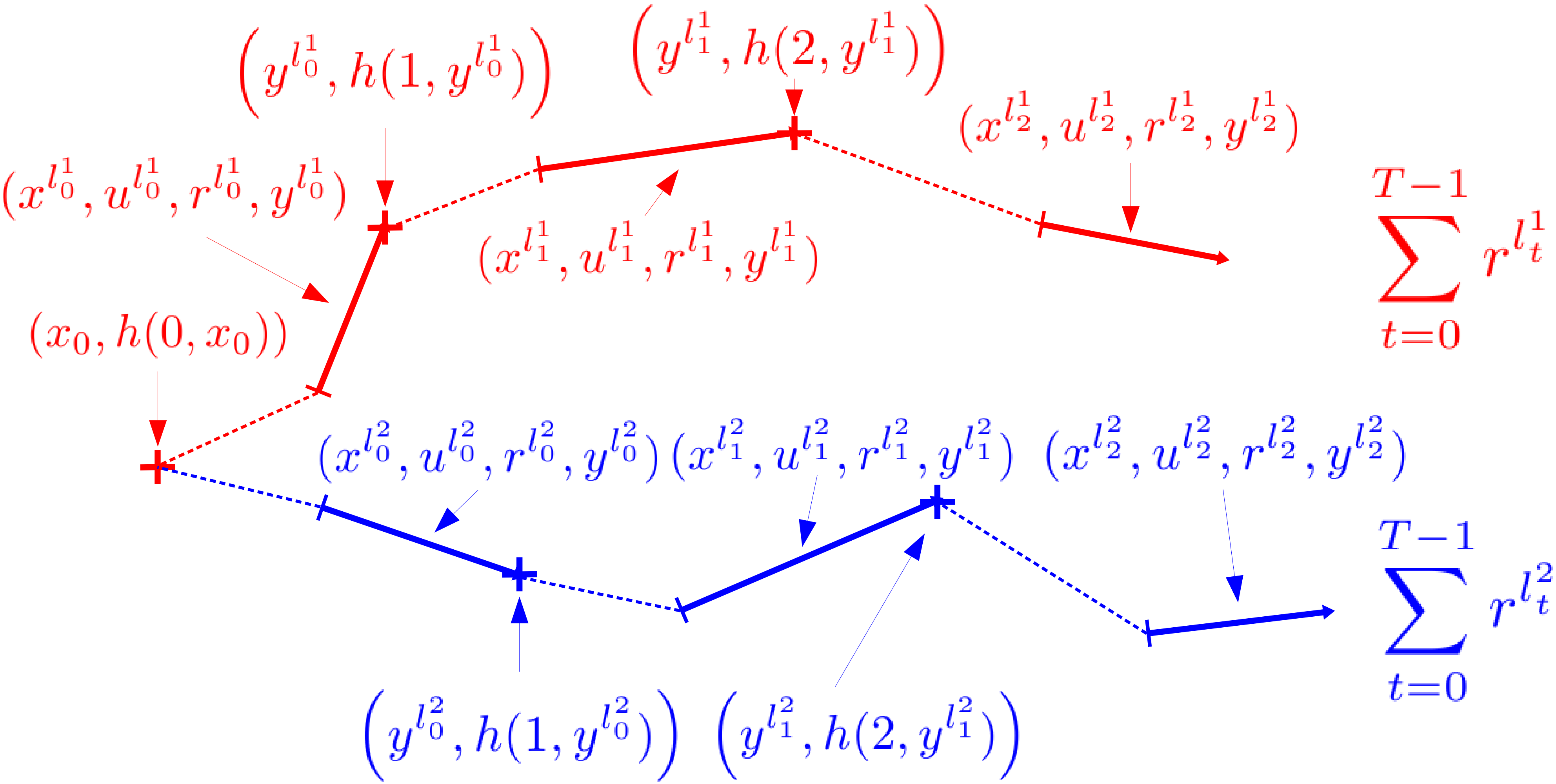
The MFMC algorithm



The MFMC algorithm



The MFMC algorithm



$$\mathfrak{M}_2^h(\mathcal{F}_8, x_0) = \frac{1}{2}\left(\sum_{t=0}^{2} r^{l_t^1} + \sum_{t=0}^{2} r^{l_t^2}\right)$$

Theoretical Analysis

Assumptions

Lipschitz continuity assumptions:

$$\exists L_f, L_\rho, L_h \in \mathbb{R}^+ : \forall\,(x, x', u, u', w) \in \mathcal{X}^2 \times \mathcal{U}^2 \times \mathcal{W},$$

$$\|f(x,u,w) - f(x',u',w)\|_{\mathcal{X}} \leq L_f(\|x - x'\|_{\mathcal{X}} + \|u - u'\|_{\mathcal{U}}),$$

$$|\rho(x,u,w) - \rho(x',u',w)| \leq L_\rho(\|x - x'\|_{\mathcal{X}} + \|u - u'\|_{\mathcal{U}}),$$

$$\forall t \in [\![0, T-1]\!], \|h(t,x) - h(t,x')\|_{\mathcal{U}} \leq L_h\|x - x'\|_{\mathcal{X}}$$

Theoretical Analysis

Assumptions

Distance metric Δ

$$\forall (x, x', u, u') \in \mathcal{X}^2 \times \mathcal{U}^2,$$
$$\Delta((x,u),(x',u')) = (\|x - x'\|_{\mathcal{X}} + \|u - u'\|_{\mathcal{U}})$$

k-dispersion

$$\alpha_k(\mathcal{P}_n) = \sup_{(x,u)\in\mathcal{X}\times\mathcal{U}} \left\{ \Delta_k^{\mathcal{P}_n}(x,u) \right\}$$

$\Delta_k^{\mathcal{P}_n}(x,u)$ denotes the distance of (x,u) to its k-th nearest neighbor (using the distance Δ) in the sample

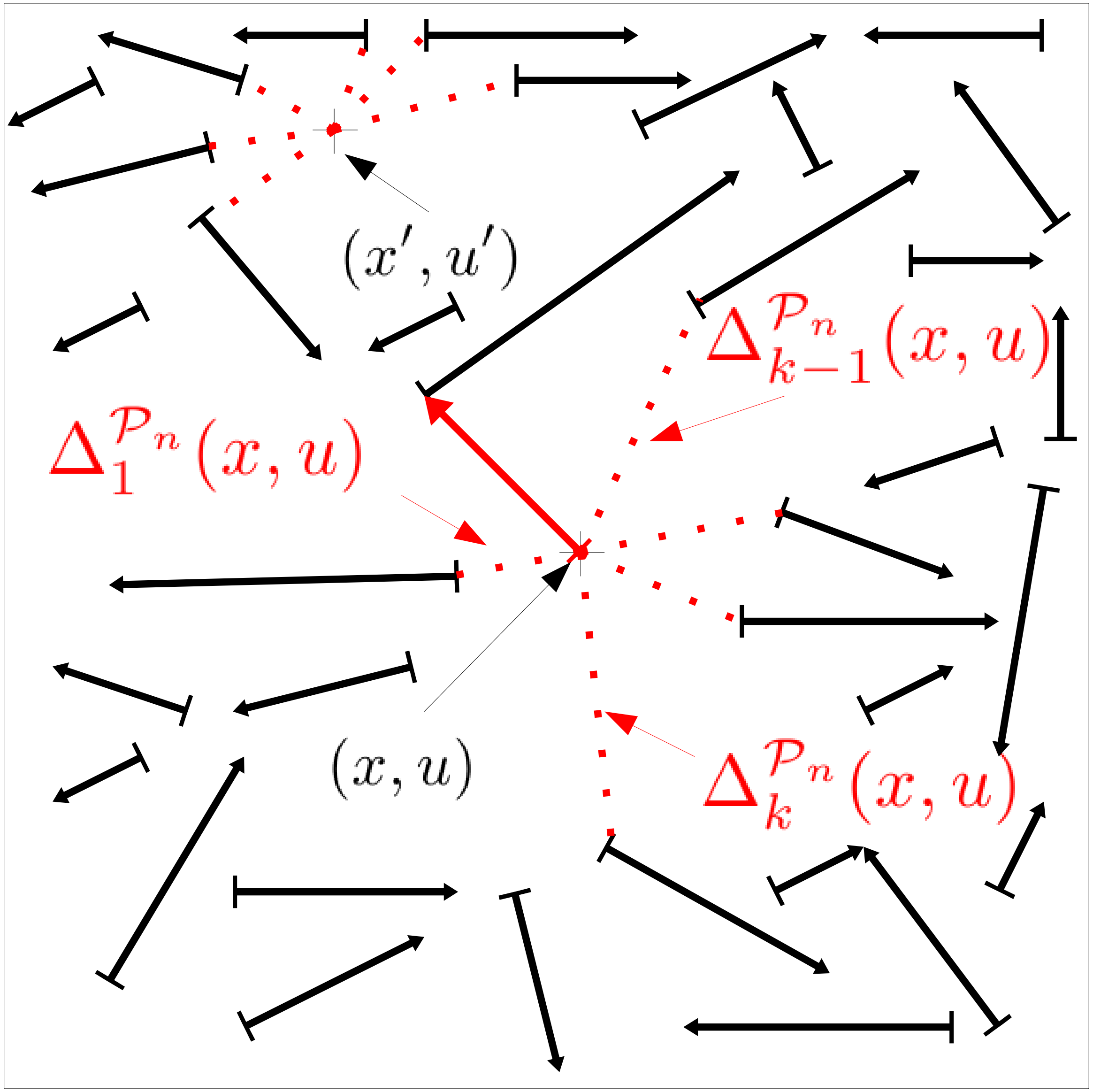
$$\mathcal{P}_n = [(x^l, u^l)]_{l=1}^n$$

Theoretical Analysis

Assumptions

The k-dispersion can be seen as the smallest radius such that all Δ-balls in X×U contain at least k elements from

$$\mathcal{P}_n = [(x^l, u^l)]_{l=1}^n$$



Theoretical Analysis

Theoretical results

Expected value of the MFMC estimator

$$E^h_{p,\mathcal{P}_n}(x_0) = \mathop{\mathbb{E}}_{w^1,\ldots,w^n \sim p_{\mathcal{W}}(.)} \left[\mathfrak{M}^h_p \left(\tilde{\mathcal{F}}_n \left(\mathcal{P}_n, w^1, \ldots, w^n \right), x_0 \right) \right]$$

Theoretical Analysis

Theoretical results

Expected value of the MFMC estimator

$$E^h_{p,\mathcal{P}_n}(x_0) = \underset{w^1,\ldots,w^n \sim p_{\mathcal{W}}(.)}{\mathbb{E}} \left[\mathfrak{M}^h_p \left(\tilde{\mathcal{F}}_n \left(\mathcal{P}_n, w^1, \ldots, w^n \right), x_0 \right) \right]$$

Theorem

$$\left| J^h(x_0) - E^h_{p,\mathcal{P}_n}(x_0) \right| \leq C \alpha_{pT} \left(\mathcal{P}_n \right)$$

with

$$C = L_\rho \sum_{t=0}^{T-1} \sum_{i=0}^{T-t-1} \left(L_f (1 + L_h) \right)^i$$

Theoretical Analysis

Theoretical results

Variance of the MFMC estimator

$$V^h_{p,\mathcal{P}_n}(x_0) = \underset{w^1,\ldots,w^n \sim p_{\mathcal{W}}(.)}{\mathbb{E}} \left[\left(\mathfrak{M}^h_p \left(\tilde{\mathcal{F}}_n \left(\mathcal{P}_n, w^1, \ldots, w^n \right), x_0 \right) - E^h_{p,\mathcal{P}_n}(x_0) \right)^2 \right]$$

Theoretical Analysis

Theoretical results

Variance of the MFMC estimator

$$V^h_{p,\mathcal{P}_n}(x_0) = \mathop{\mathbb{E}}_{w^1,\ldots,w^n \sim p_{\mathcal{W}}(.)} \left[\left(\mathfrak{M}^h_p \left(\tilde{\mathcal{F}}_n \left(\mathcal{P}_n, w^1, \ldots, w^n \right), x_0 \right) - E^h_{p,\mathcal{P}_n}(x_0) \right)^2 \right]$$

Theorem

$$V^h_{p,\mathcal{P}_n}(x_0) \leq \left(\frac{\sigma_{R^h}(x_0)}{\sqrt{p}} + 2C\alpha_{pT}\left(\mathcal{P}_n\right) \right)^2$$

with

$$C = L_\rho \sum_{t=0}^{T-1} \sum_{i=0}^{T-t-1} \left(L_f (1 + L_h) \right)^i$$

Experimental Illustration

Benchmark

Dynamics:

$$x_{t+1} = \sin\left(\frac{\pi}{2}(x_t + u_t + w_t)\right)$$

Reward function:

$$\rho(x_t, u_t, w_t) = \frac{1}{2\pi} e^{-\frac{1}{2}(x_t^2 + u_t^2)} + w_t$$

Policy to evaluate:

$$h(t, x) = -\frac{x}{2}, \qquad \forall x \in \mathcal{X}, \forall t \in \{0, \ldots, T-1\}$$

$$\mathcal{X} = [-1, 1], \mathcal{U} = [-\tfrac{1}{2}, \tfrac{1}{2}], \mathcal{W} = [-\tfrac{\epsilon}{2}, \tfrac{\epsilon}{2}] \text{ with } \epsilon = 0.1$$

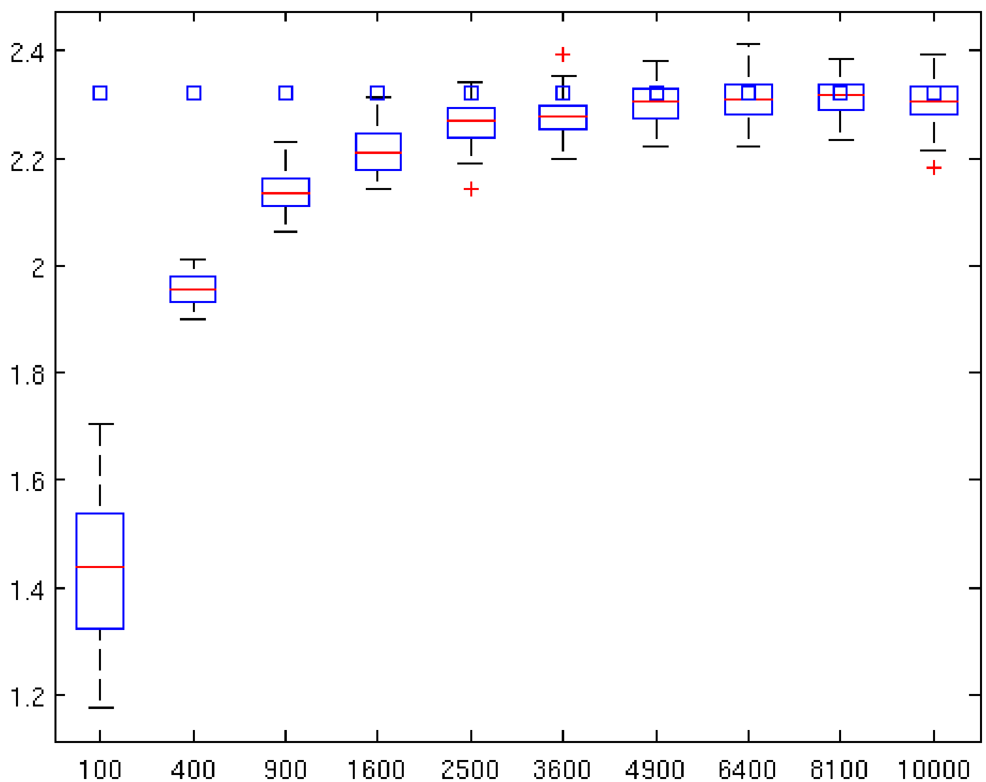
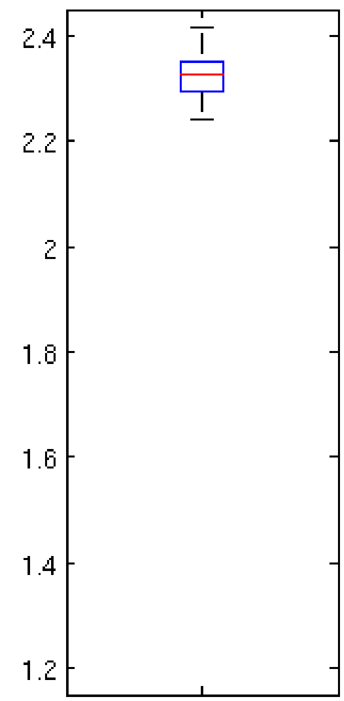
Other information:

$p_W(.)$ is uniform

Experimental Illustration

Influence of n

Simulations for p = 10, n = 100 … 10 000, uniform grid, T = 15, x_0 = - 0.5

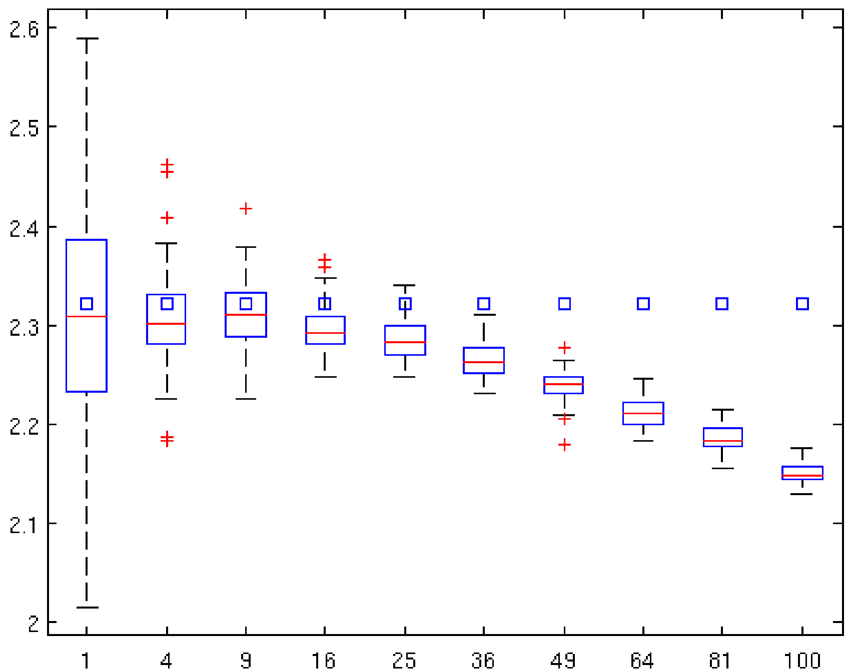
Model-free Monte Carlo estimator	Monte Carlo estimator
	
n = 100 … 10 000, p = 10	p = 10

Experimental Illustration

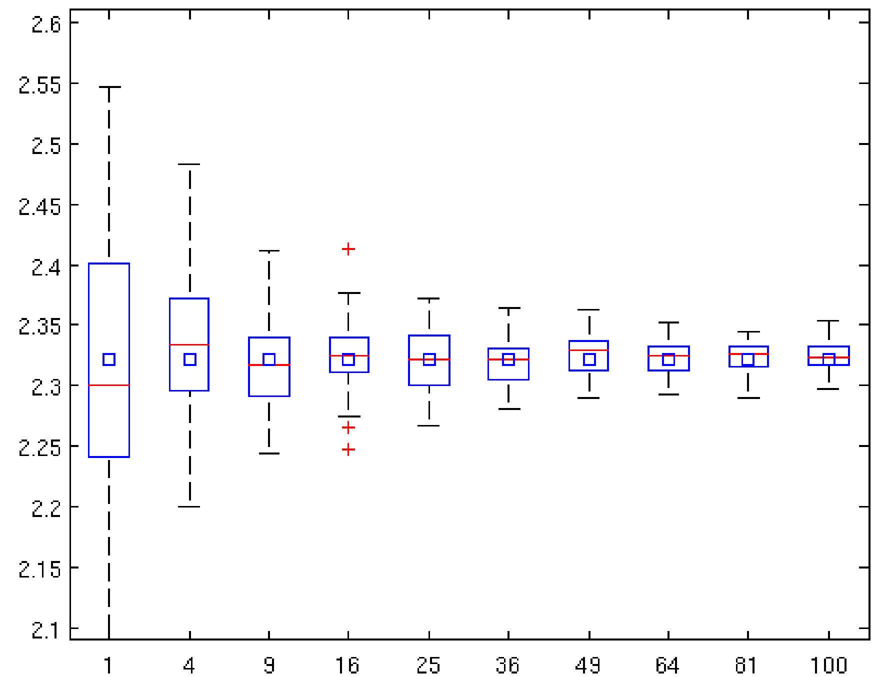
Influence of p

Simulations for p = 1 … 100, n = 10 000, uniform grid, T = 15, x_0 = - 0.5

Model-free Monte Carlo estimator	Monte Carlo estimator



p = 1 … 100, n=10 000

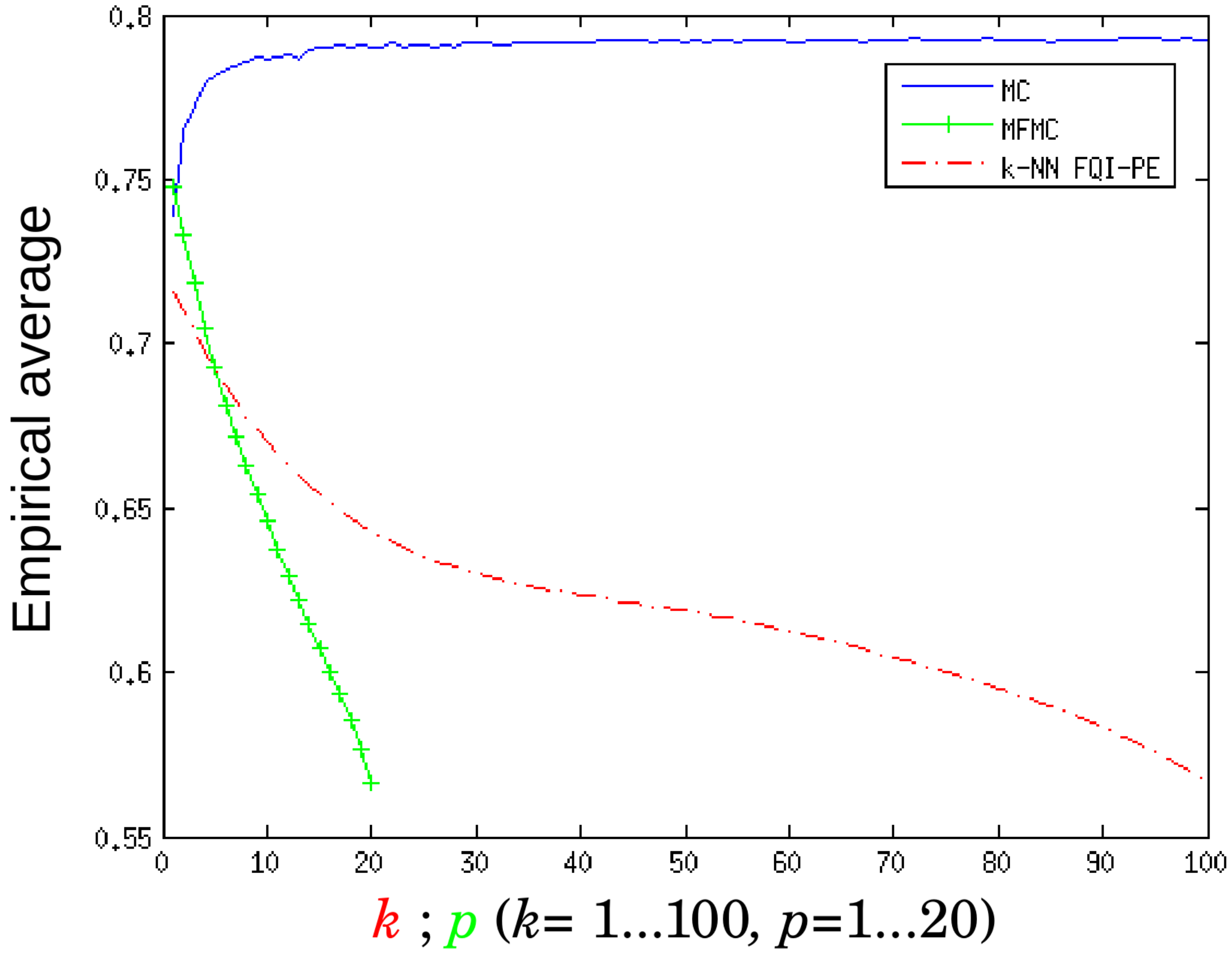


p = 1 … 100

Experimental Illustration

MFMC vs FQI-PE

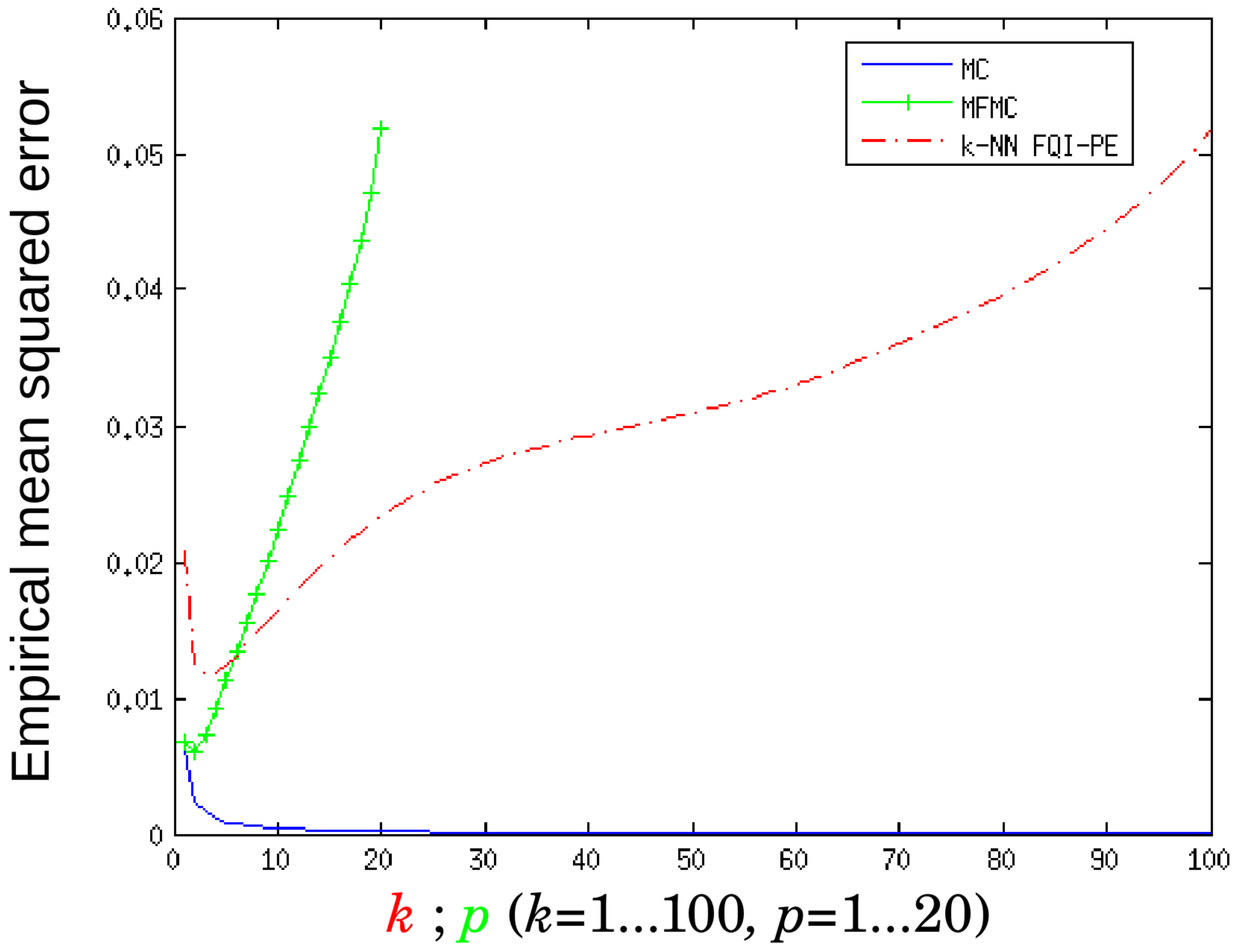
Comparison with the FQI-PE algorithm using k-NN, n=100, T=5 .



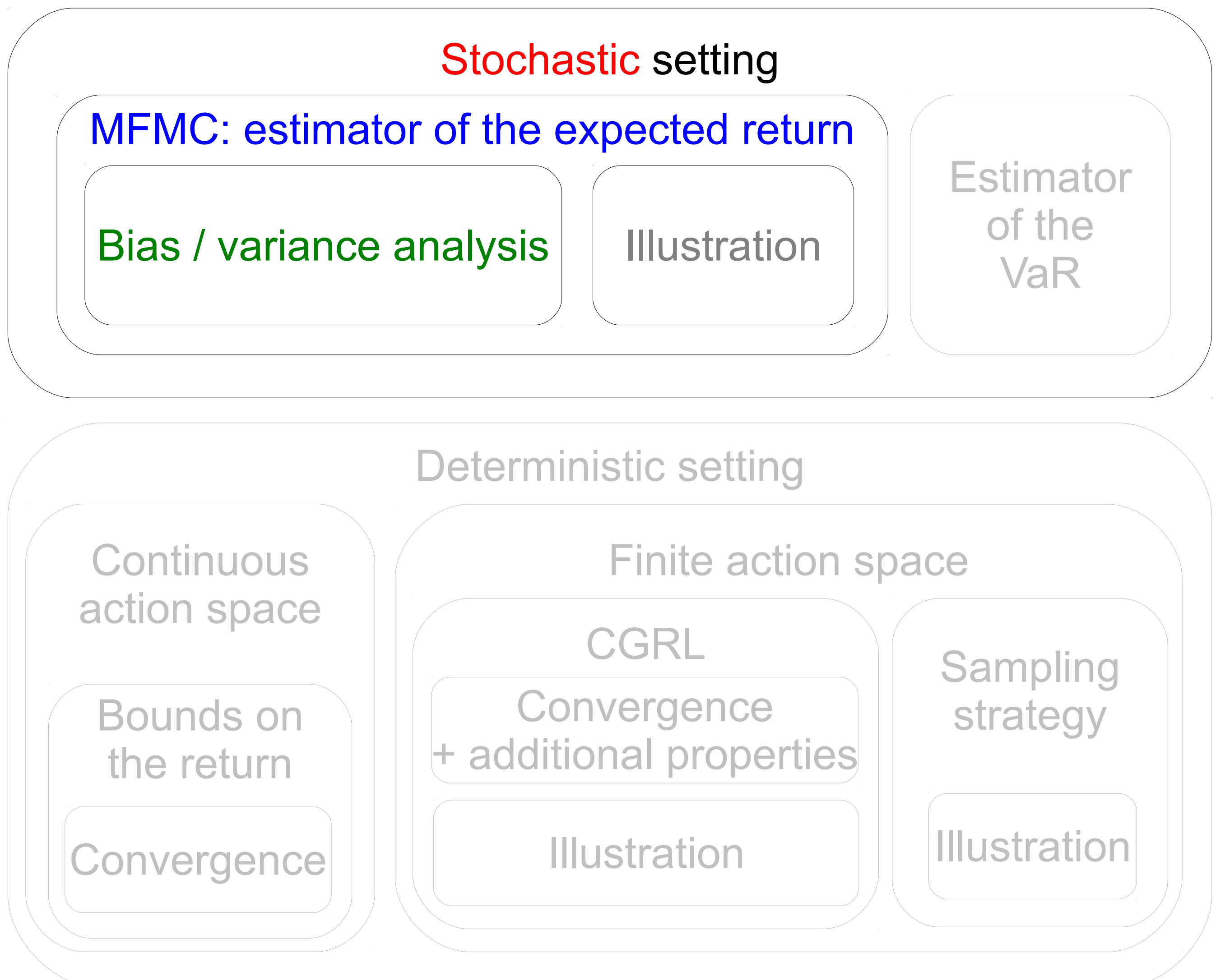
Experimental Illustration

MFMC vs FQI-PE

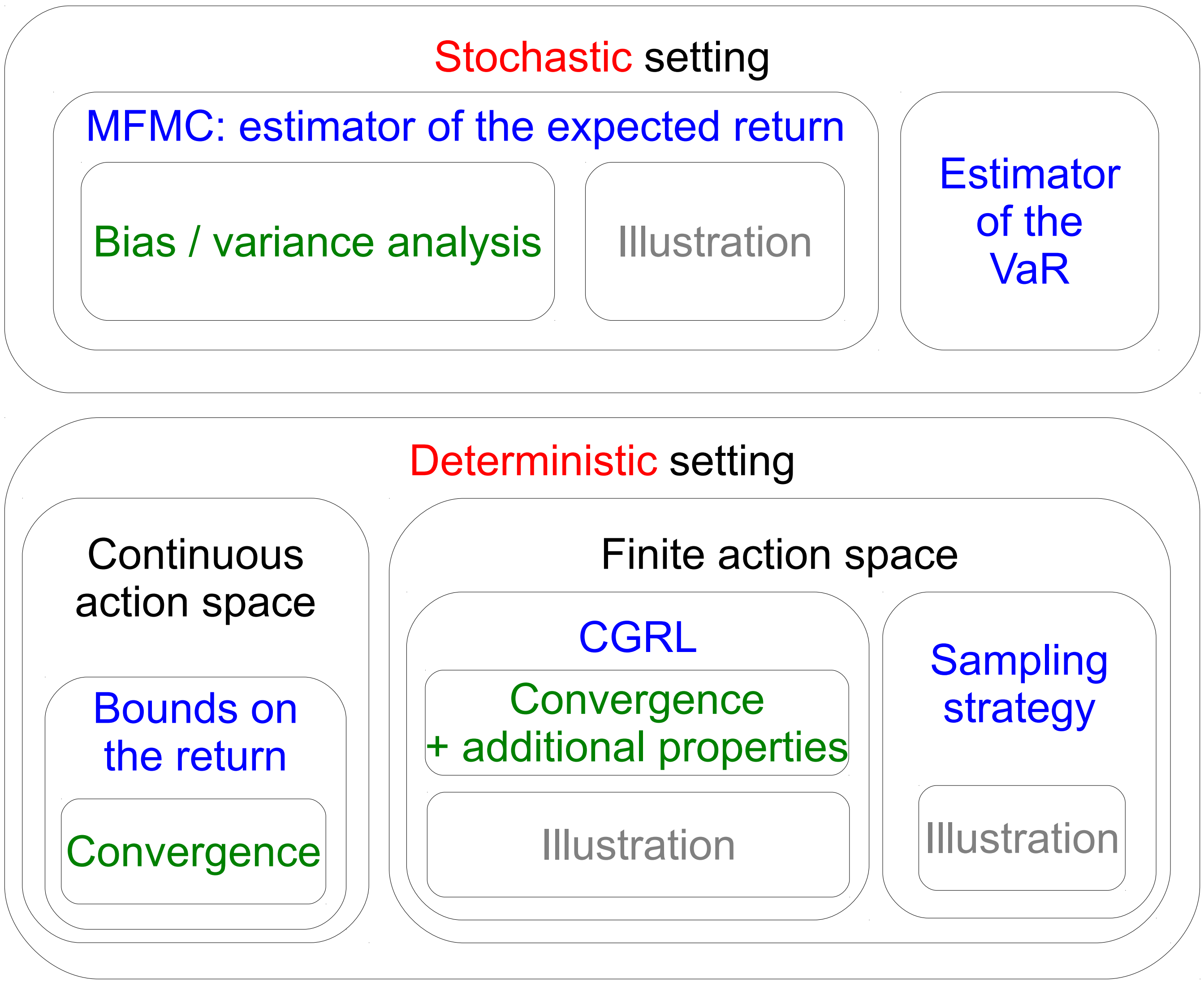
Comparison with the FQI-PE algorithm using k-NN, n=100, T=5 .



Research map



Research map



Estimating the Performances of Policies

Risk-sensitive criterion

Consider again the *p* artificial trajectories that were rebuilt by the MFMC estimator. The Value-at-Risk of the policy *h*

$$J_{RS}^{h,(b,c)}(x_0) = \begin{cases} -\infty & \text{if } P\left(R^h(x_0, w_0, \ldots, w_{T-1}) < b\right) > c \\ J^h(x_0) & \text{otherwise} \end{cases}$$

can be straightforwardly estimated as follows:

$$\tilde{J}_{RS}^{h,(b,c)}(x_0) = \begin{cases} -\infty & \text{if } \frac{1}{p}\sum_{i=1}^{p} \mathbb{I}_{\{\mathbf{r}^i < b\}} > c \ , \\ \mathfrak{M}^h\left(\mathcal{F}_n, x_0\right) & \text{otherwise} \end{cases}$$

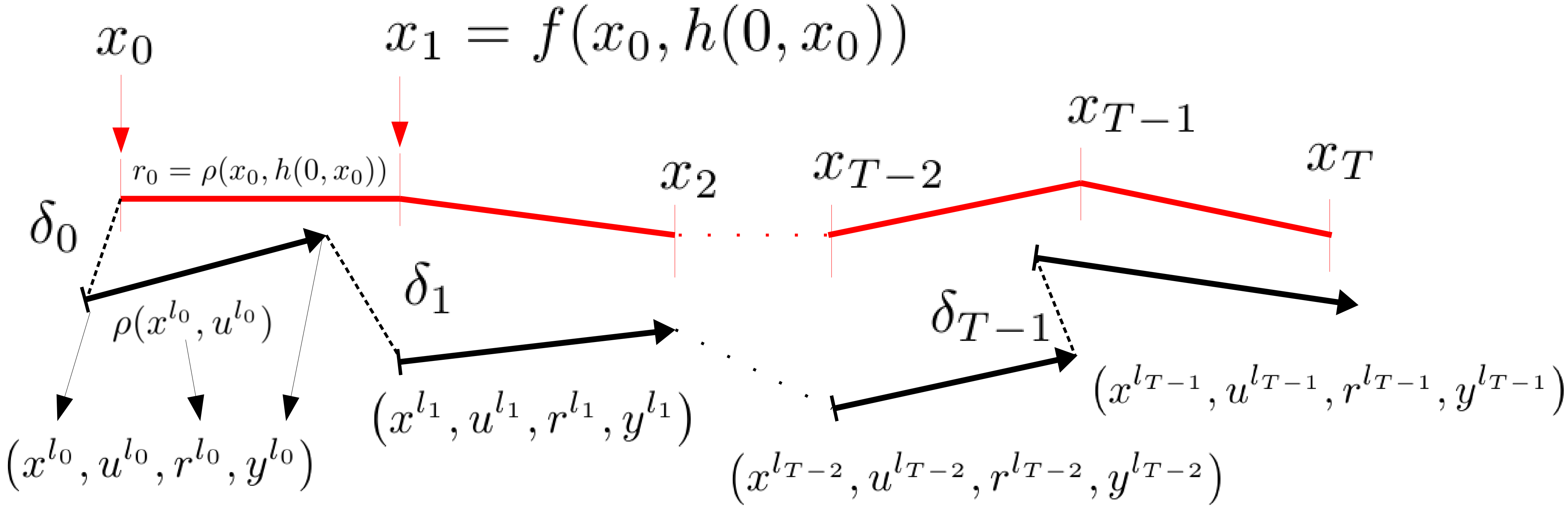
with $$\mathbf{r}^i = \sum_{t=0}^{T-1} r^{l_t^i}$$

$c \in [0,1[\quad b \in \mathbb{R}$

Deterministic Case: Computing Bounds

Bounds from a Single Trajectory

Given an artificial trajectory : $\tau = \left[\left(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t}\right)\right]_{t=0}^{T-1}$



$$\delta_0 = \left\|x^{l_0} - x_0\right\|_{\mathcal{X}} + \left\|u^{l_0} - h(0, x_0)\right\|_{\mathcal{U}} \ ,$$

$$\delta_1 = \left\|y^{l_0} - x^{l_1}\right\|_{\mathcal{X}} + \left\|u^{l_1} - h(1, y^{l_0})\right\|_{\mathcal{U}} \quad \ldots$$

Deterministic Case: Computing Bounds

Bounds from a Single Trajectory

Proposition:

Let $\left[\left(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t}\right)\right]_{t=0}^{T-1}$ be an artificial trajectory. Then,

$$J^h(x_0) \geq \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta\left((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t})\right)$$

with

$$L_{Q_{T-t}} = L_\rho \sum_{i=0}^{T-t-1} \left(L_f\left(1 + L_h\right)\right)^i$$

$$y^{l_{-1}} = x_0$$

Deterministic Case: Computing Bounds

Maximal Bounds

Maximal lower and upper-bounds

$$L^h(\mathcal{F}_n, x_0) = \max_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1} \in \mathcal{F}_n^T} \sum_{t=0}^{T-1} r^{l_t} - \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta\left((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t}) \right)$$

$$U^h(\mathcal{F}_n, x_0) = \min_{[(x^{l_t}, u^{l_t}, r^{l_t}, y^{l_t})]_{t=0}^{T-1} \in \mathcal{F}_n^T} \sum_{t=0}^{T-1} r^{l_t} + \sum_{t=0}^{T-1} L_{Q_{T-t}} \Delta\left((y^{l_{t-1}}, h(t, y^{l_{t-1}})), (x^{l_t}, u^{l_t}) \right)$$

Deterministic Case: Computing Bounds

Tightness of Maximal Bounds

Proposition:

$$\exists C_b > 0 : \quad J^h(x_0) - L^h(\mathcal{F}_n, x_0) \leq C_b \alpha_1(\mathcal{P}_n)$$
$$U^h(\mathcal{F}_n, x_0) - J^h(x_0) \leq C_b \alpha_1(\mathcal{P}_n)$$

Inferring Safe Policies

From Lower Bounds to Cautious Policies

Consider the set of open-loop policies:

$$\Pi = \{\pi : \{0, \ldots, T-1\} \to \mathcal{U}\}$$

For such policies, bounds can be computed in a similar way

We can then search for a specific policy for which the associated lower bound is maximized:

$$\hat{\pi}^*_{\mathcal{F}_n, x_0} \in \arg\max_{\pi \in \Pi} \quad L^\pi(\mathcal{F}_n, x_0)$$

A O($T\,n$ ²) algorithm for doing this: the CGRL algorithm (Cautious approach to Generalization in RL)

Inferring Safe Policies

Convergence

Theorem

Let $\mathfrak{J}^*(x_0)$ be the set of optimal open-loop policies:

$$\mathfrak{J}^*(x_0) = \arg\max_{\pi \in \Pi} J^{\pi}(x_0) \ ,$$

and let us suppose that $\mathfrak{J}^*(x_0) \neq \Pi$ (if $\mathfrak{J}^*(x_0) = \Pi$, the search for an optimal policy is indeed trivial). We define

$$\epsilon(x_0) = \min_{\pi \in \Pi \setminus \mathfrak{J}^*(x_0)} \left\{ \left(\max_{\pi' \in \Pi} J^{\pi'}(x_0) \right) - J^{\pi}(x_0) \right\} \ .$$

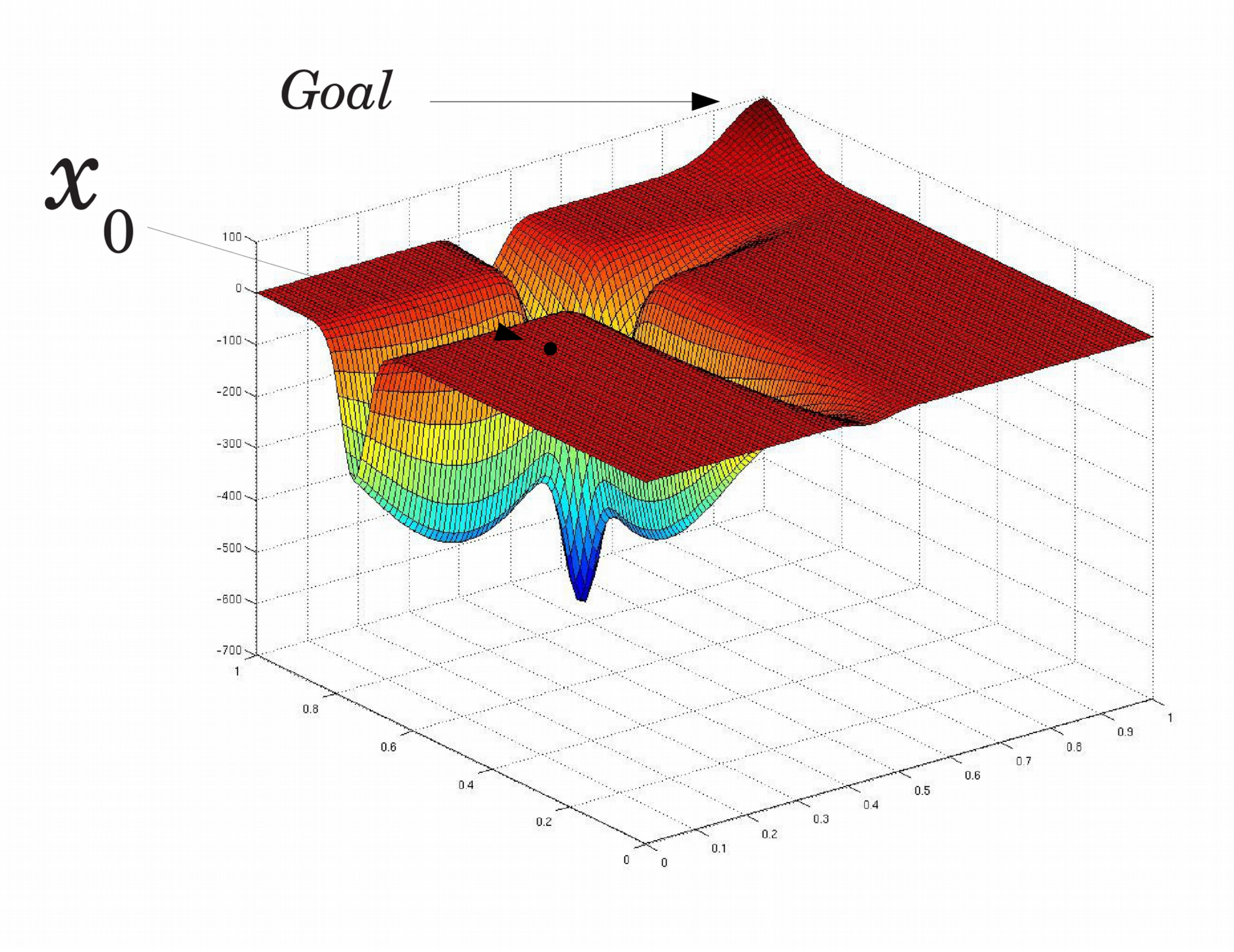
Then,

$$\left(C_b' \alpha^*(\mathcal{P}_n) < \epsilon(x_0) \right) \implies \hat{\pi}^*_{\mathcal{F}_n, x_0} \in \mathfrak{J}^*(x_0) \ .$$

Inferring Safe Policies

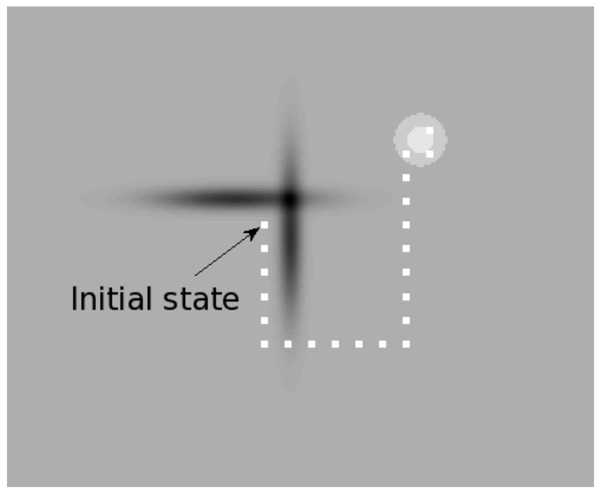
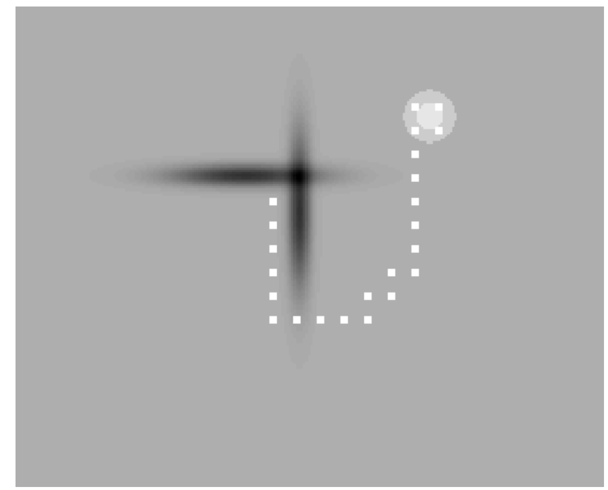
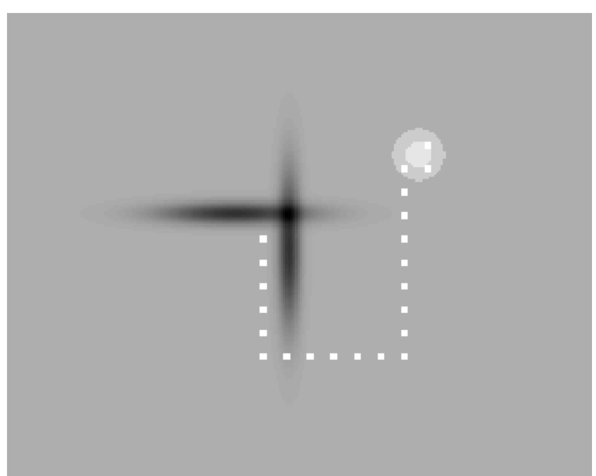
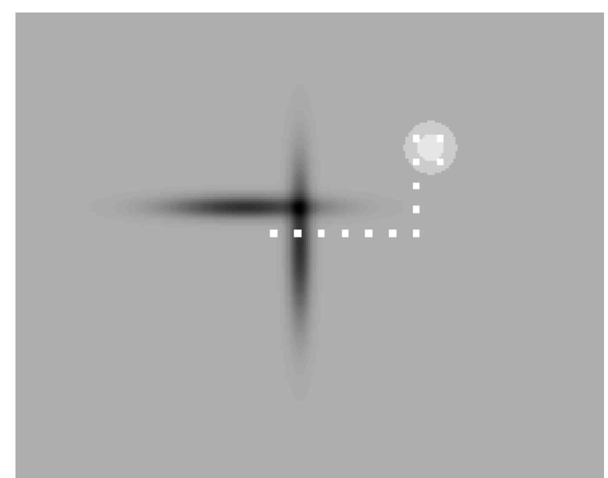
Experimental Results

- The puddle world benchmark



Inferring Safe Policies

Experimental Results

	CGRL	FQI (Fitted Q Iteration)
The state space is uniformly covered by the sample		
Information about the Puddle area is removed		

Inferring Safe Policies

Bonus

Theorem

Let $\pi^*_{x_0} \in \mathfrak{J}^*(x_0)$ *be an optimal open-loop policy. Let us assume that one can find in* $\mathcal{F}_n$ *a sequence of* T *one-step system transitions*

$$\left[\left(x^{l_0}, u^{l_0}, r^{l_0}, x^{l_1}\right), \left(x^{l_1}, u^{l_1}, r^{l_1}, x^{l_2}\right), \ldots, \left(x^{l_{T-1}}, u^{l_{T-1}}, r^{l_{T-1}}, x^{l_T}\right)\right] \in \mathcal{F}_n^T$$

such that

$$\begin{aligned} x^{l_0} &= x_0 \ , \\ u^{l_t} &= \pi^*_{x_0}(t) \qquad \forall t \in \{0, \ldots, T-1\} \ . \end{aligned}$$

Let $\hat{\pi}^*_{\mathcal{F}_n, x_0}$ *be such that*

$$\hat{\pi}^*_{\mathcal{F}_n, x_0} \in \underset{\pi \in \Pi}{\arg\max} \qquad L^{\pi}(\mathcal{F}_n, x_0) \ .$$

Then,

$$\hat{\pi}^*_{\mathcal{F}_n, x_0} \in \mathfrak{J}^*(x_0) \ .$$

Sampling Strategies

An Artificial Trajectories Viewpoint

Given a sample of system transitions

$$\mathcal{F}_n = \left\{ \left(x^l, u^l, r^l, y^l \right) \in \mathcal{X} \times \mathcal{U} \times \mathbb{R} \times \mathcal{X} \right\}_{l=1}^{n}$$

How can we determine where to sample additional transitions ?

We define the set of candidate optimal policies:

$$\Pi(\mathcal{F}, x_0) = \left\{ \pi \in \Pi \quad \middle| \quad \forall \pi' \in \Pi, U^{\pi}(\mathcal{F}, x_0) \geq L^{\pi'}(\mathcal{F}, x_0) \right\}$$

A transition $(x, u, r, y) \in \mathcal{X} \times \mathcal{U} \times \mathbb{R} \times \mathcal{X}$ is compatible with $\mathcal{F}$ if

$$\forall (x^l, u^l, r^l, y^l) \in \mathcal{F}, \quad (u^l = u) \implies \begin{cases} \left| r - r^l \right| & \leq L_\rho \| x - x^l \|_{\mathcal{X}} \\ \left\| y - y^l \right\|_{\mathcal{X}} & \leq L_f \| x - x^l \|_{\mathcal{X}} \end{cases}$$

and we denote by $\mathcal{C}(\mathcal{F})$ the set of all such compatible transitions.

Sampling Strategies

An Artificial Trajectories Viewpoint

Iterative scheme:

$$(x^{m+1}, u^{m+1}) \in \underset{(x,u)\in\mathcal{X}\times\mathcal{U}}{\arg\min} \left\{ \max_{\substack{(r,y)\in\mathbb{R}\times\mathcal{X}\ s.t.(x,u,r,y)\in\mathcal{C}(\mathcal{F}_m) \\ \pi\in\mathit{\Pi}(\mathcal{F}_m\cup\{(x,u,r,y)\},x_0)}} \delta^\pi\left(\mathcal{F}_m\cup\{(x,u,r,y)\},x_0\right) \right\}$$

with

$$\delta^\pi(\mathcal{F}, x_0) = U^\pi(\mathcal{F}, x_0) - L^\pi(\mathcal{F}, x_0)$$

Conjecture:

$$\exists m_0 \in \mathbb{N}\setminus\{0\} : \forall m \in \mathbb{N}, \Big(m \geq m_0\Big) \implies \mathit{\Pi}\left(\mathcal{F}_m, x_0\right) = \mathfrak{J}^*(x_0)$$

Sampling Strategies

Illustration

Action space: $\mathcal{U} = \{-0.20, -0.10, 0, +0.10, +0.20\}$

Dynamics and reward function:

$$f(x, u) = x + u$$

$$\rho(x, u) = x + u$$

Horizon: $T = 3$

Initial sate:

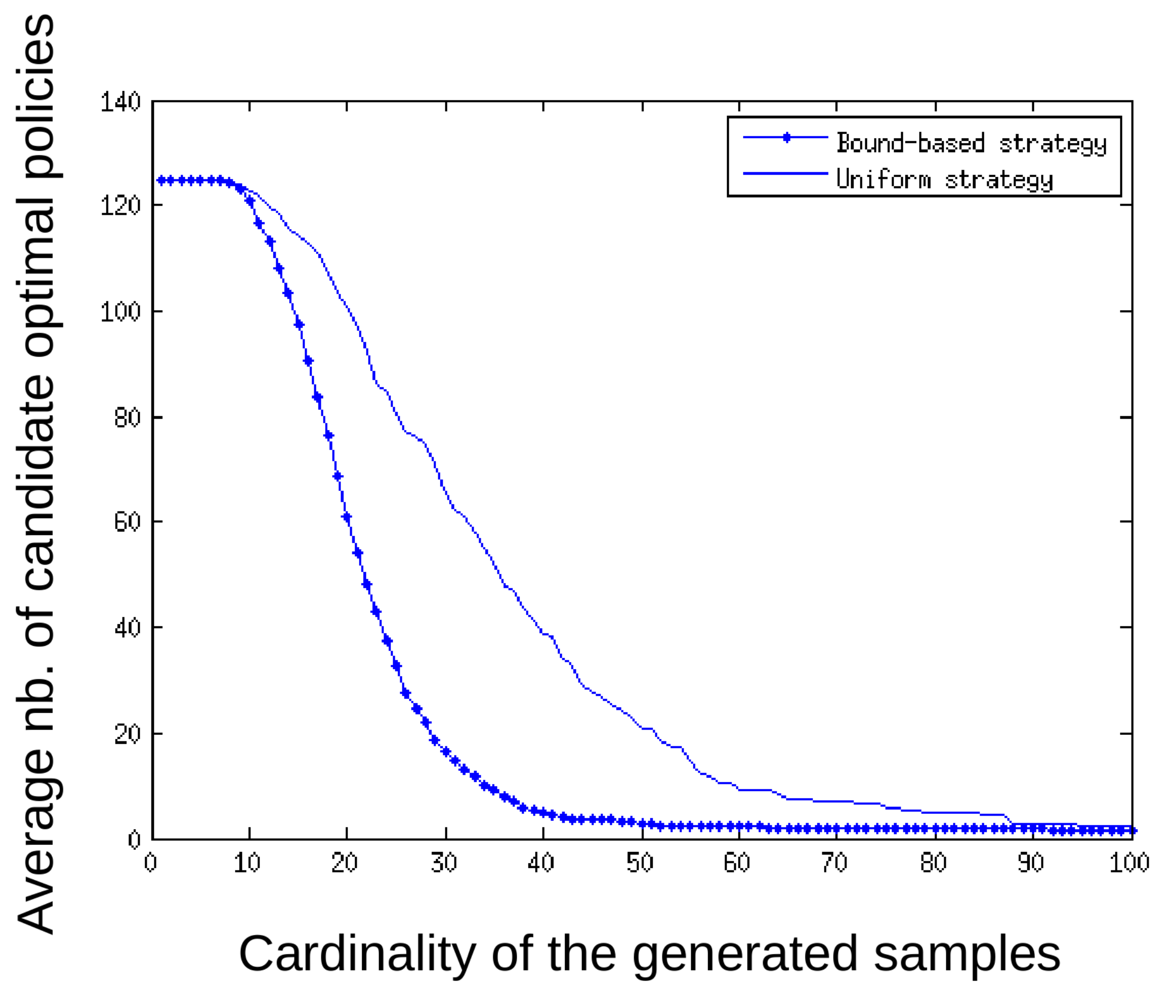
$$x_0 = -0.65$$

Total number of policies:

$$5^3 = 125$$

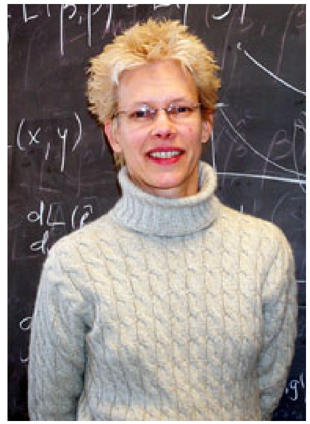
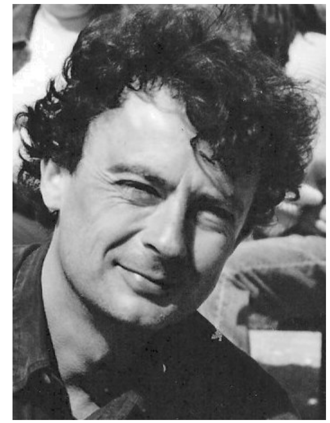
Number of transitions needed for discriminating:

$$5 + 25 + 125 = 155$$



Thank you

References

"**Batch mode reinforcement learning based on the synthesis of artificial trajectories**". R. Fonteneau, S.A. Murphy, L. Wehenkel and D. Ernst. Annals of Operations Research,Volume 208, Issue 1, pp 383-416, 2013.

"**Generating informative trajectories by using bounds on the return of control policies**". R. Fonteneau, S.A. Murphy, L. Wehenkel and D. Ernst. Proceedings of the Workshop on Active Learning and Experimental Design 2010 (in conjunction with AISTATS 2010), 2-page highlight paper, Chia Laguna, Sardinia, Italy, May 16, 2010.

"**Model-free Monte Carlo-like policy evaluation**". R. Fonteneau, S.A. Murphy, L. Wehenkel and D. Ernst. In Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2010), JMLR W&CP 9, pp 217-224, Chia Laguna, Sardinia, Italy, May 13-15, 2010.

"**A cautious approach to generalization in reinforcement learning**". R. Fonteneau, S.A. Murphy, L. Wehenkel and D. Ernst. Proceedings of The International Conference on Agents and Artificial Intelligence (ICAART 2010), 10 pages, Valencia, Spain, January 22-24, 2010.

"**Inferring bounds on the performance of a control policy from a sample of trajectories**". R. Fonteneau, S.A. Murphy, L. Wehenkel and D. Ernst. In Proceedings of The IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL 2009), 7 pages, Nashville, Tennessee, USA, 30 March-2 April, 2009.