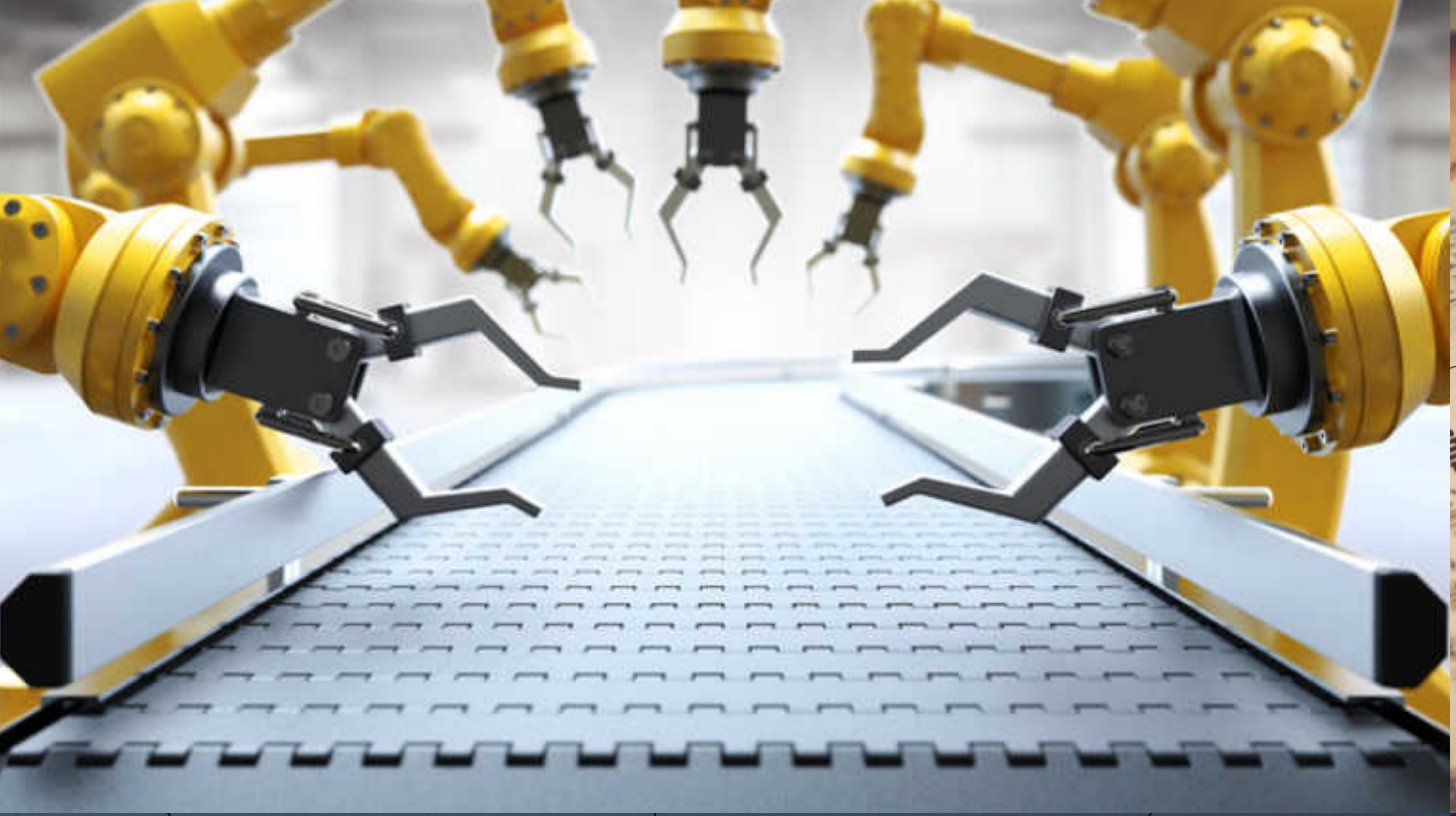
A Machine Learning Approach to Online Optimization

PRINCETON UNIVERSITY ORFE

joint work with
Dimitris Bertsimas

MIT MANAGEMENT SLOAN SCHOOL

Bartolomeo Stellato — INFORMS Annual Meeting 2020




REDAKTIONSSCHLUSS DIESER AUSGABE: 21.45 UHR
DOW JONES
EURO in DOLLAR
BANTLEON
ANZEIGE
UNTERNEHMEN
ONLINE-FINANZTOOL:

Existing approaches

Machine learning for optimization

[Bengio et al. (2020)]

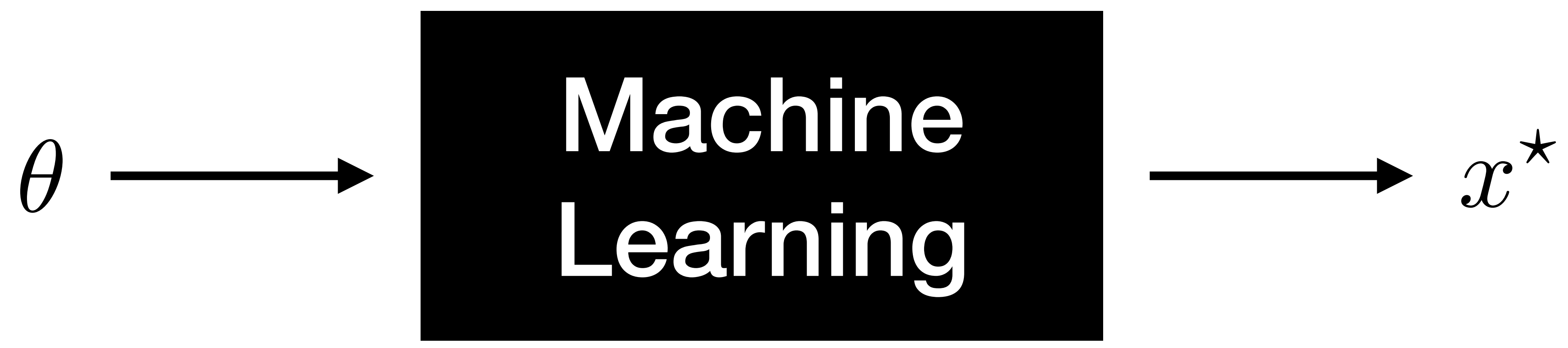
$\theta \longrightarrow$ **Machine Learning** $\longrightarrow x^\star$

Combinatorial Optimization [Smith (1999)]
TSP [Vinyals (2017)]

Existing approaches

Machine learning for optimization

[Bengio et al. (2020)]



Combinatorial Optimization [Smith (1999)]
TSP [Vinyals (2017)]

Very small problems

Imprecise

Existing approaches

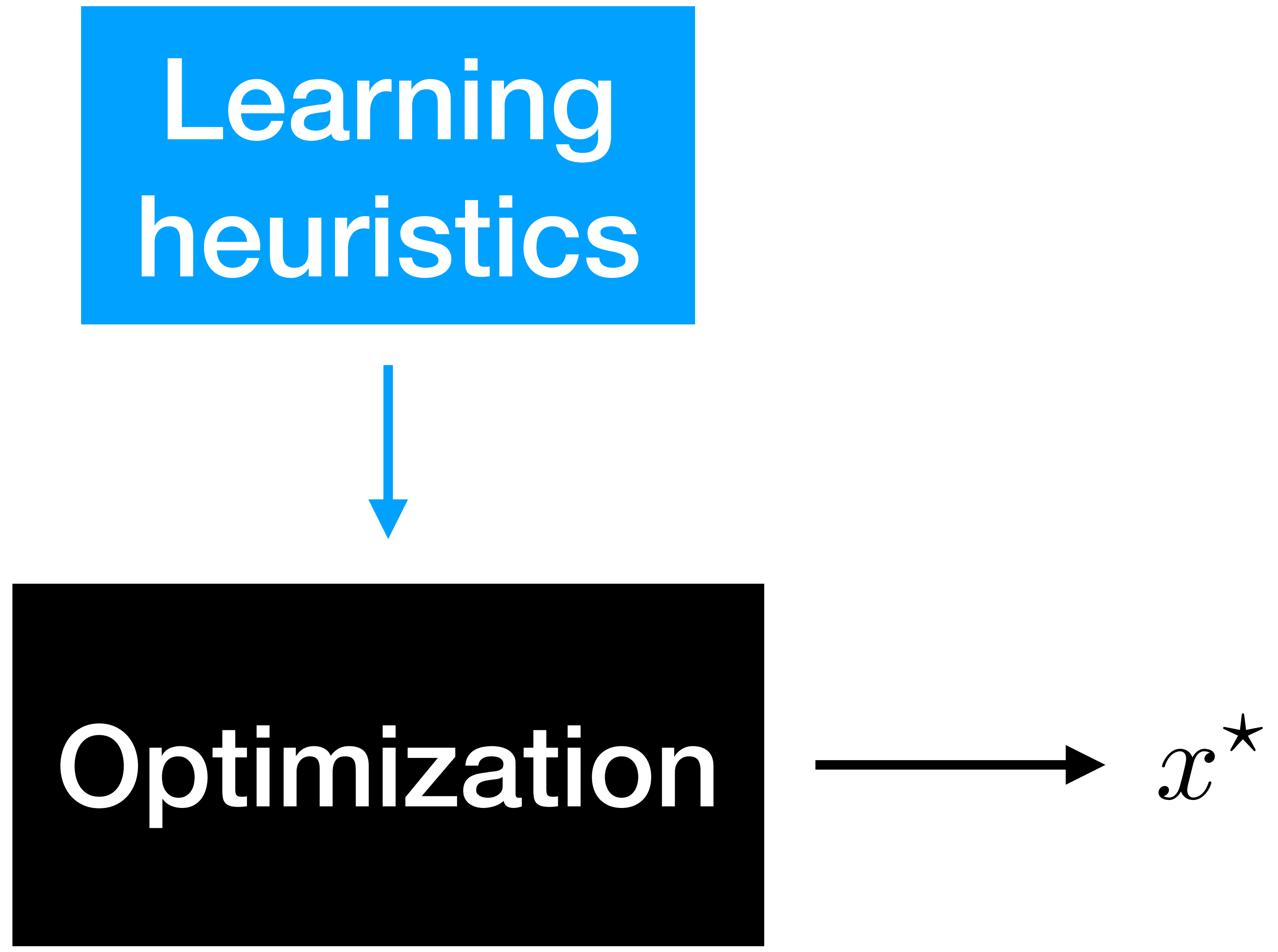
Machine learning for optimization

[Bengio et al. (2020)]

Combinatorial Optimization [Smith (1999)]
TSP [Vinyals (2017)]

Very small problems

Imprecise



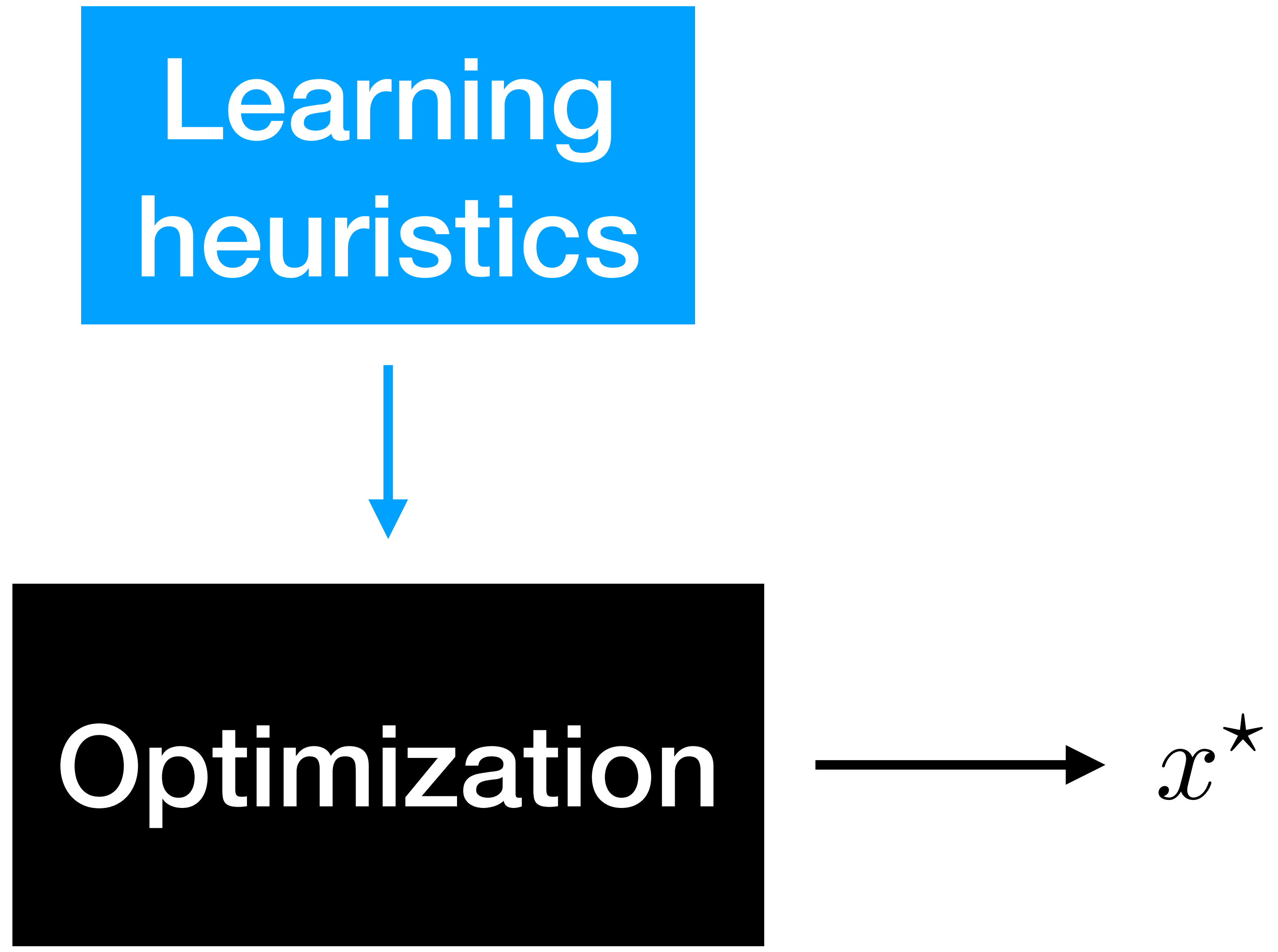
Branch and Bound [Khalil et al. (2016)]
Decompositions [Bonami et a. (2018)]
TSP [Khalil et al (2017)]
Cutting planes [Tang et al. (2020)]
Many more….

Existing approaches

Machine learning for optimization

[Bengio et al. (2020)]

Combinatorial Optimization [Smith (1999)]
TSP [Vinyals (2017)]



Branch and Bound [Khalil et al. (2016)]
Decompositions [Bonami et a. (2018)]
TSP [Khalil et al (2017)]
Cutting planes [Tang et al. (2020)]
Many more….

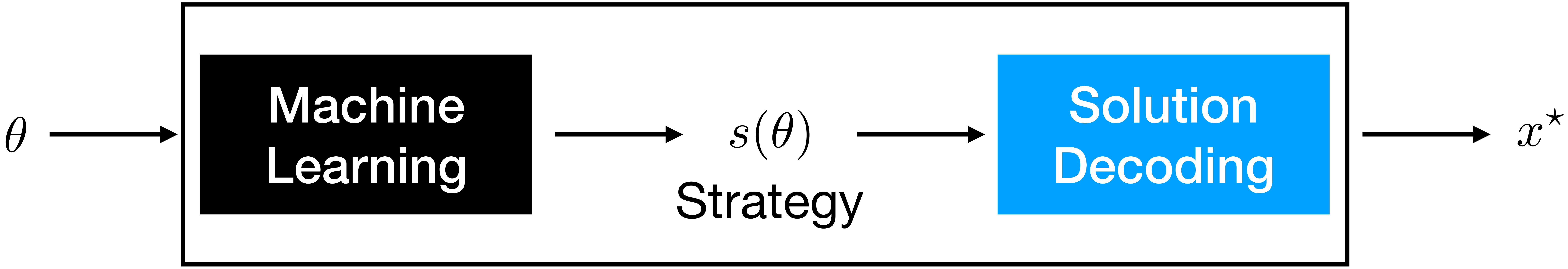
Very small problems

Imprecise

Still hard to solve

Does not focus on parametric problems

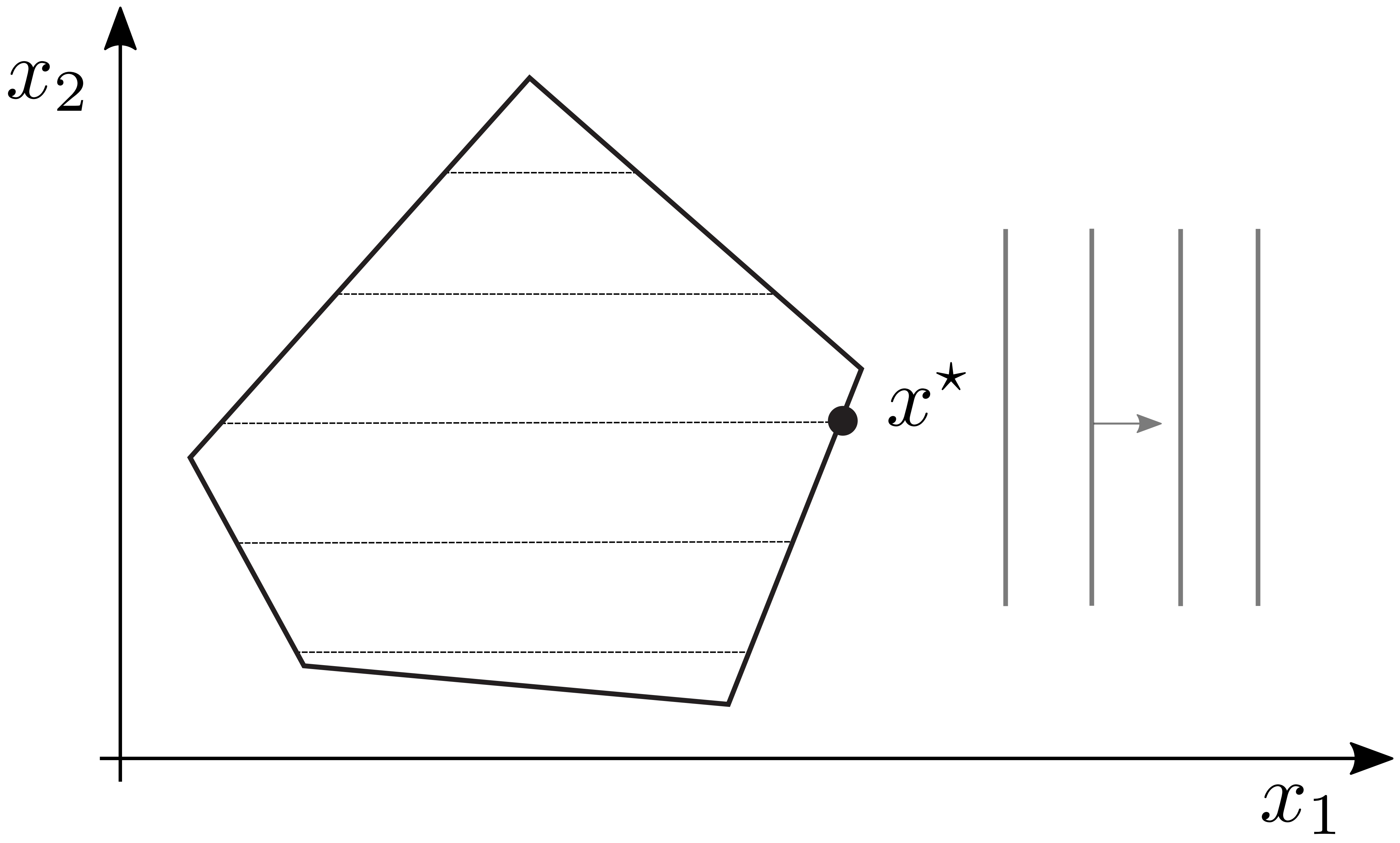
Machine learning optimizer



Strategies for mixed-integer optimization

$$\begin{array}{ll} \text{minimize} & c(\theta)^T x \\ \text{subject to} & A(\theta)x \leq b(\theta) \\ & x_{\mathcal{I}} \in \mathbf{Z}^d \end{array}$$

Parameters

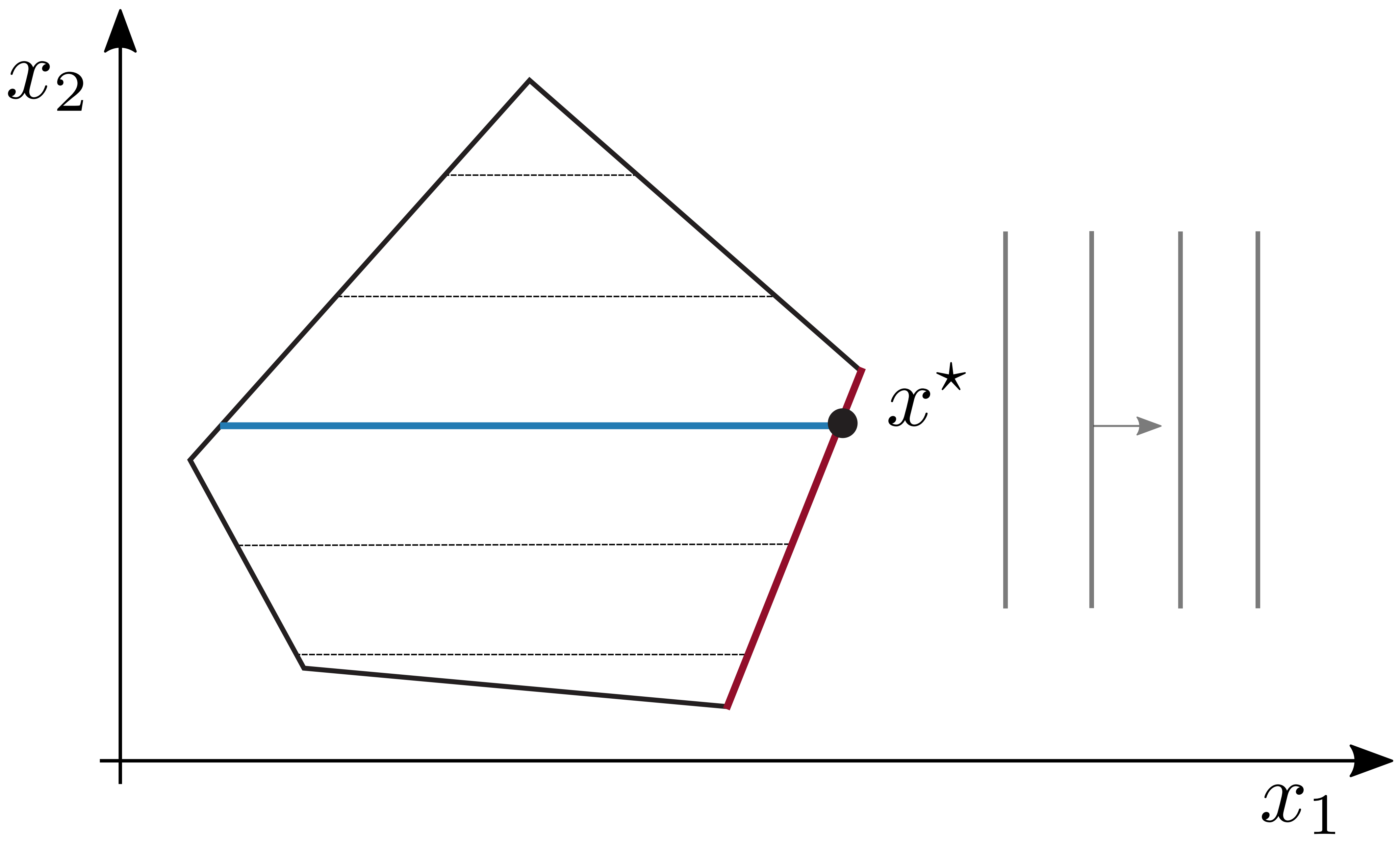


Strategies for mixed-integer optimization

$$s(\theta) = (\mathcal{T}(\theta), x^\star_{\mathcal{I}}(\theta))$$

Tight constraints

Integer variables



Computing the solution from the strategy

$$s(\theta) \longrightarrow \boxed{\text{Solution Decoding}} \longrightarrow x^\star$$

$$\begin{array}{ll} \text{minimize} & c(\theta)^T x \\ \text{subject to} & A(\theta)x \leq b(\theta) \\ & x_{\mathcal{I}} \in \mathbf{Z}^d \end{array}$$

Computing the solution from the strategy

$s(\theta) \longrightarrow$ **Solution Decoding** $\longrightarrow x^\star$

Convex optimization

$$\begin{array}{ll} \text{minimize} & c(\theta)^T x \\ \text{subject to} & A(\theta)x \leq b(\theta) \\ & x_{\mathcal{I}} \in \mathbf{Z}^d \end{array} \quad \xrightarrow{s(\theta)} \quad \begin{array}{ll} \text{minimize} & c(\theta)^T x \\ \text{subject to} & A_i(\theta)x = b_i(\theta), \quad \forall i \in \mathcal{T}(\theta) \\ & x_{\mathcal{I}} = x_{\mathcal{I}}^\star(\theta) \end{array}$$

Computing the solution from the strategy

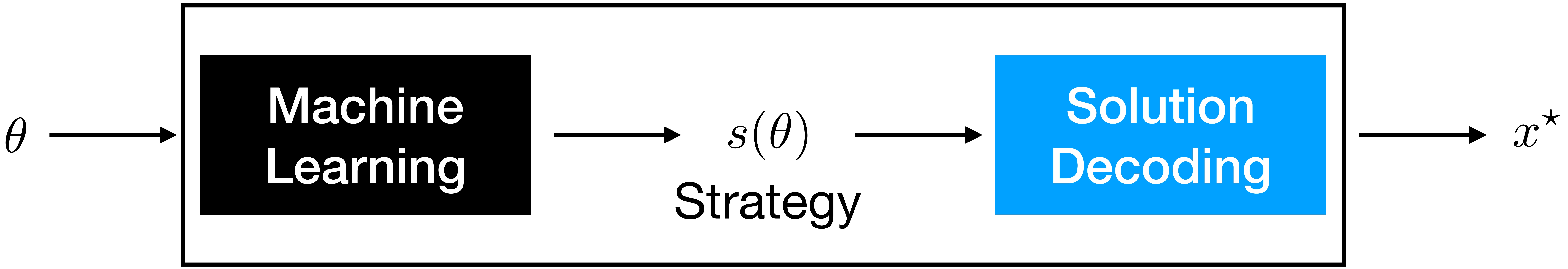
$s(\theta) \longrightarrow$ **Solution Decoding** $\longrightarrow x^\star$

Convex optimization

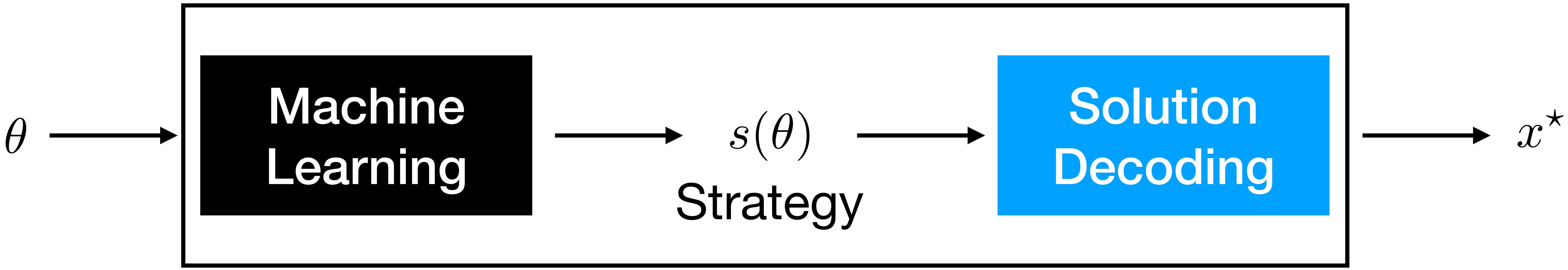
$$\begin{array}{ll} \text{minimize} & c(\theta)^T x \\ \text{subject to} & A(\theta)x \leq b(\theta) \\ & x_{\mathcal{I}} \in \mathbf{Z}^d \end{array} \quad \xrightarrow{s(\theta)} \quad \begin{array}{ll} \text{minimize} & c(\theta)^T x \\ \text{subject to} & A_i(\theta)x = b_i(\theta), \quad \forall i \in \mathcal{T}(\theta) \\ & x_{\mathcal{I}} = x^\star_{\mathcal{I}}(\theta) \end{array}$$

KKT Linear system

Predicting the strategies



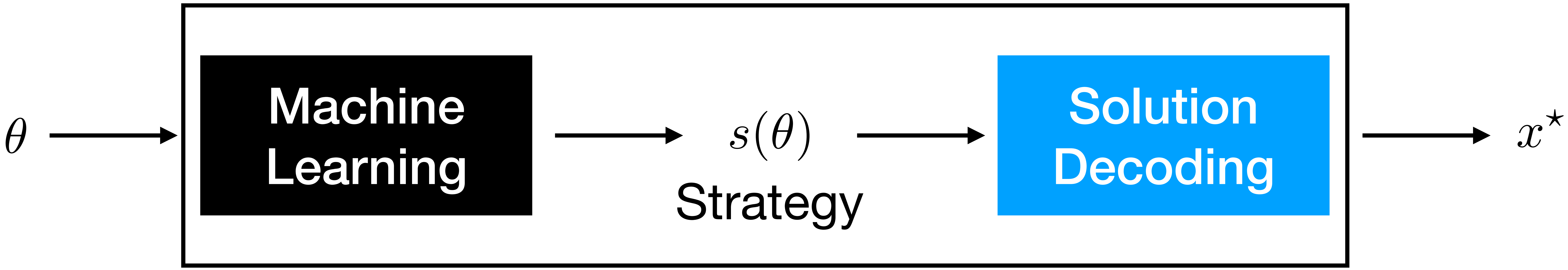
Predicting the strategies



N data $(\theta_i, s(\theta_i))$

M labels (strategies) $\mathcal{S}$

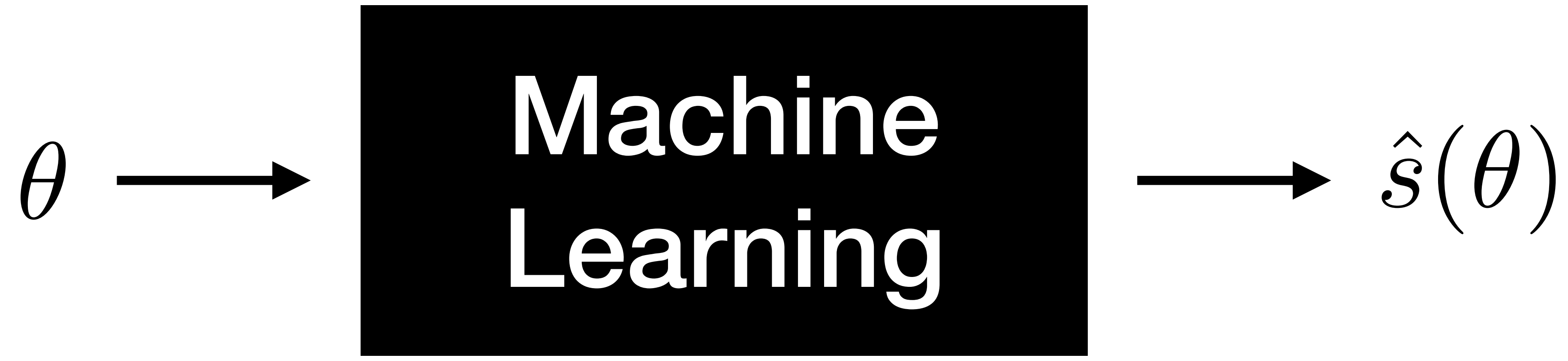
Predicting the strategies



Multiclass classification

N data $(\theta_i, s(\theta_i))$

M labels (strategies) $\mathcal{S}$



Predicting the strategies

$\theta \longrightarrow$ **Machine Learning** $\longrightarrow s(\theta) \longrightarrow$ **Solution Decoding** $\longrightarrow x^\star$

Strategy

Multiclass classification

N data $(\theta_i, s(\theta_i))$

M labels (strategies) $\mathcal{S}$

$\theta \longrightarrow$ **Machine Learning** $\longrightarrow \hat{s}(\theta)$

Neural Networks

PyTorch

What happens if we have many strategies?

Multiclass classification

$\theta \longrightarrow$ **Machine Learning** $\longrightarrow \hat{s}(\theta)$

N data $(\theta_i, s(\theta_i))$

M labels (strategies) $\mathcal{S}$ **(Too many!)**

What happens if we have many strategies?

Multiclass classification

$\theta \longrightarrow$ Machine Learning $\longrightarrow \hat{s}(\theta)$

N data $(\theta_i, s(\theta_i))$

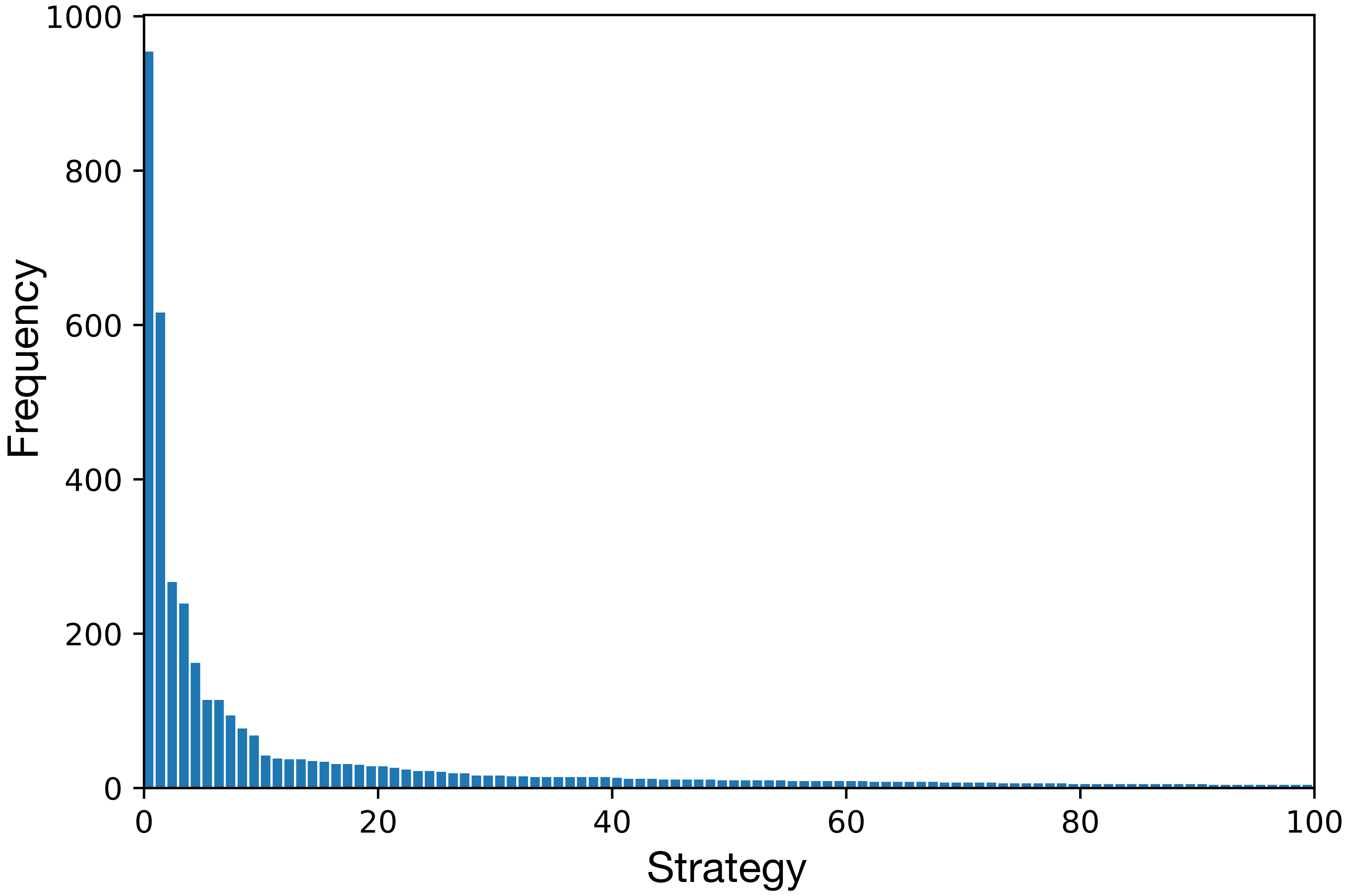
M labels (strategies) $\mathcal{S}$ **(Too many!)**

IDEA

Can we discard some strategies?

Strategy pruning

Keep only most frequent strategies



Example

$N = 5035$ samples
$M = 835$ strategies

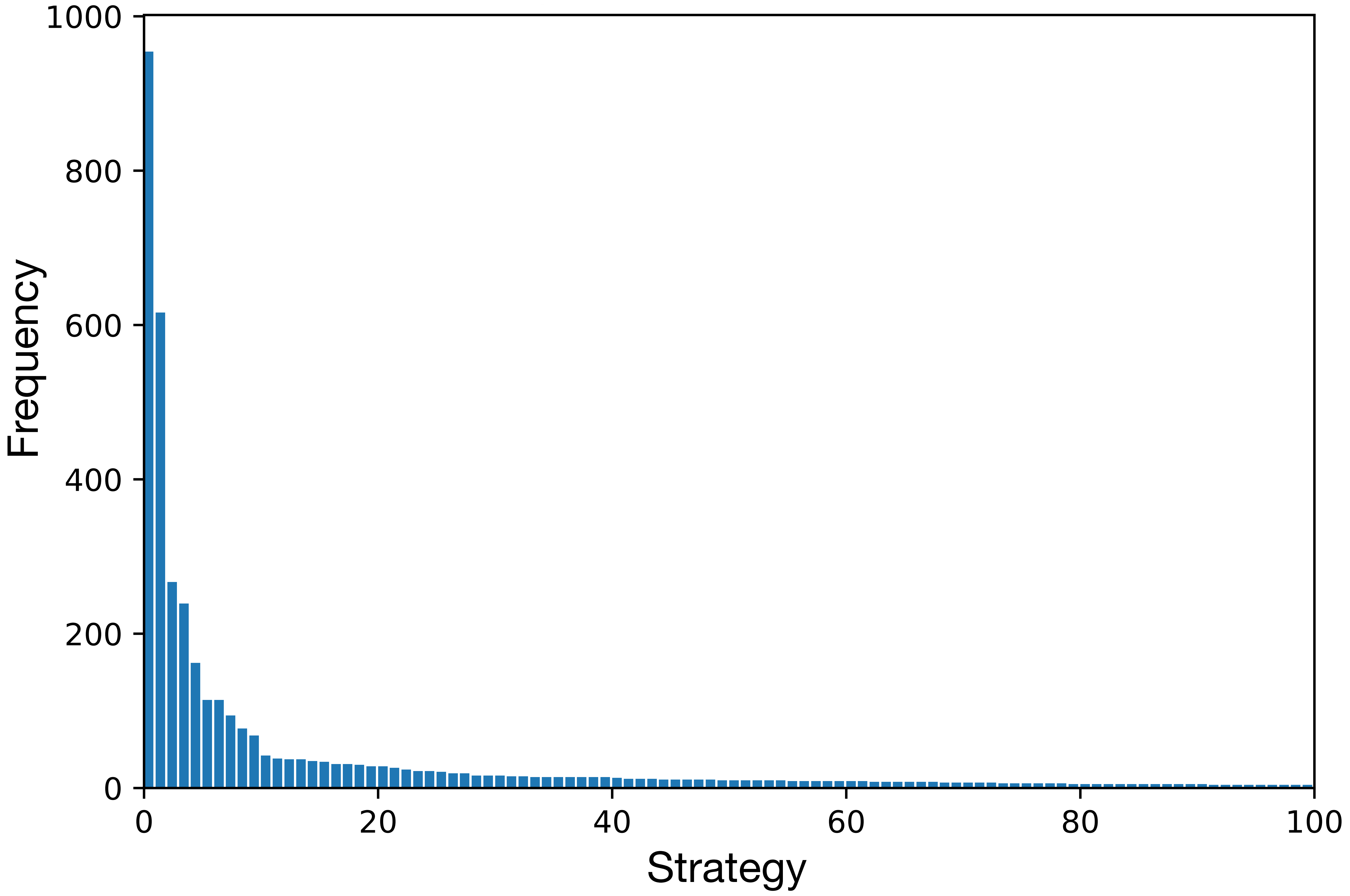
Just a few really matter

First 100 cover 76.4% samples

First 200 cover 82.9% samples

Strategy pruning

Keep only most frequent strategies



Example

$N = 5035$ samples
$M = 835$ strategies

Just a few really matter

First 100 cover 76.4% samples

First 200 cover 82.9% samples

Pruning

- Keep strategies for $> 80\%$ samples
- Reassign other samples

Optimal pruning
is a MILO (too slow)

Iterative pruning

Pseudocode

initialize $\alpha = 0.2, \epsilon = 0.02$
for $k = 1, \dots, k_{\max}$ **do**

- Select *most frequent strategies* $\mathcal{S}_\alpha$ for at least $1 - \alpha\%$ samples
- Reassign *discarded samples* θ_i to strategies in $\mathcal{S}_\alpha$.
- If $\max_i(\mathrm{subopt}(\theta_i)) < \epsilon$ **break**
- $\alpha \leftarrow \alpha/2$

Iterative pruning

Pseudocode

initialize $\alpha = 0.2, \epsilon = 0.02$
for $k = 1, \ldots, k_{\max}$ **do**

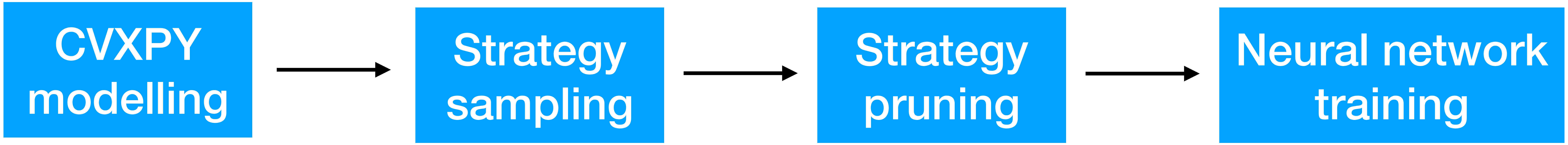
- Select *most frequent strategies* $\mathcal{S}_\alpha$ for at least $1 - \alpha\%$ samples
- Reassign *discarded samples* θ_i to strategies in $\mathcal{S}_\alpha$.
- If $\max_i(\mathrm{subopt}(\theta_i)) < \epsilon$ **break**
- $\alpha \leftarrow \alpha/2$

No need to reassign all the samples

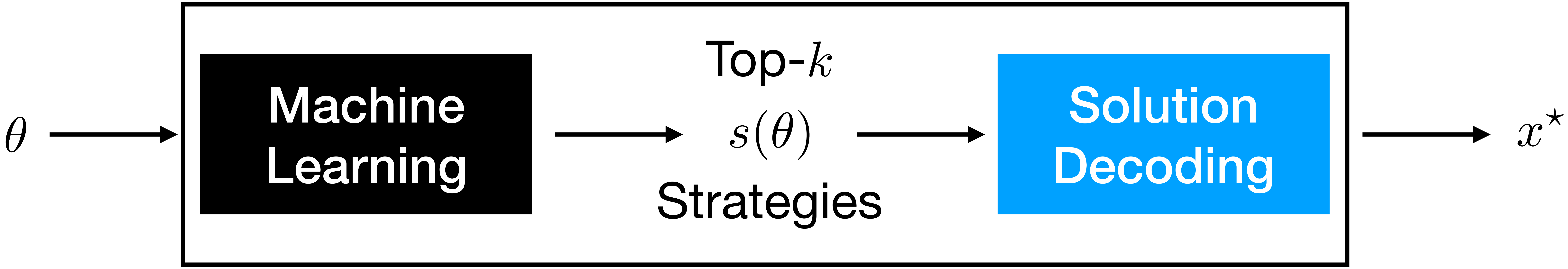
MLOPT: Machine Learning Optimizer

`github.com/bstellato/mlopt`

Offline learning

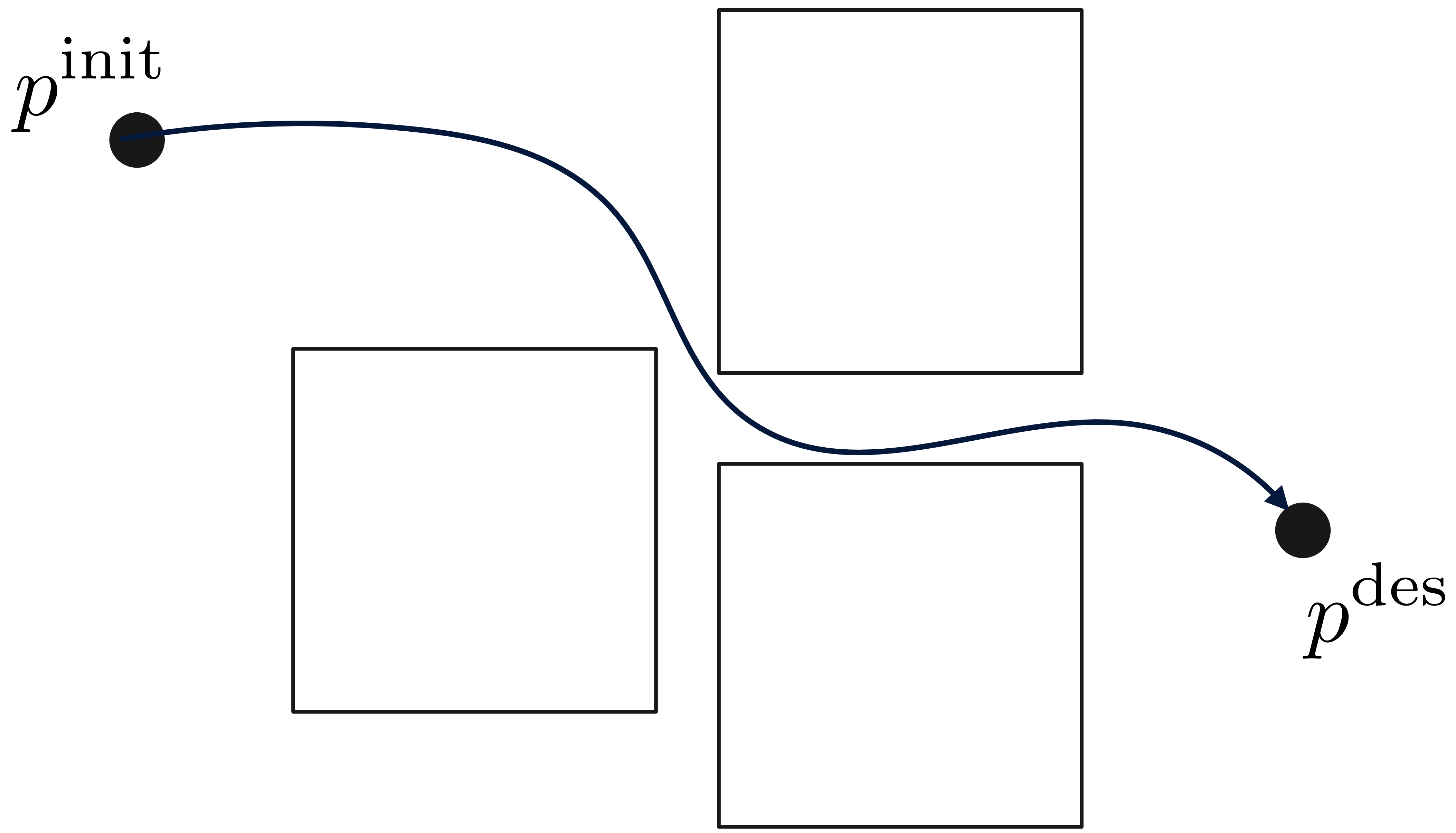


Fast online predictions



Example

Motion planning with obstacles



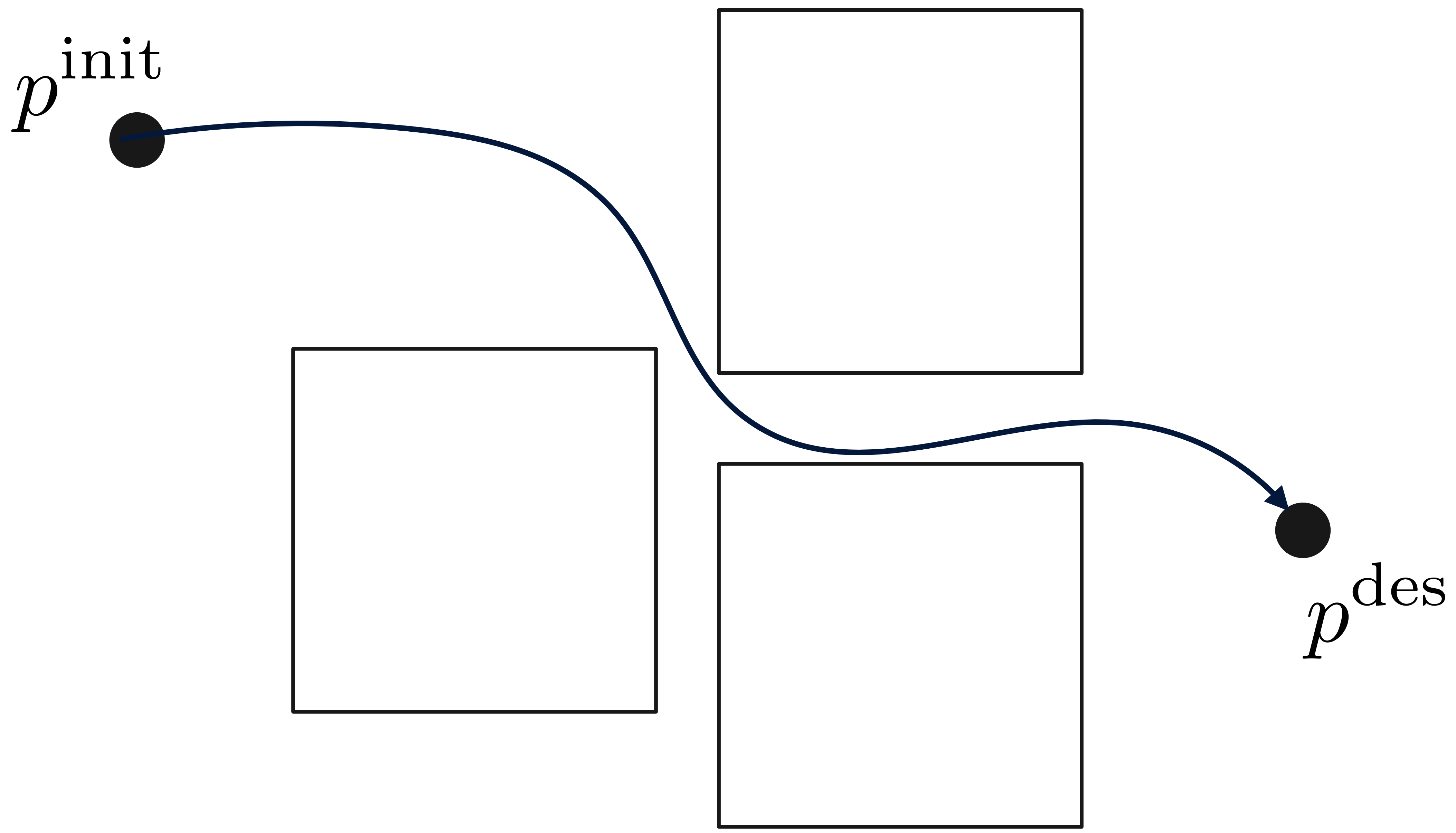
p^{init} initial position
v^{init} initial velocity

p_t position
v_t velocity

p^{des} desired position

Example

Motion planning with obstacles



p^{init} initial position
v^{init} initial velocity

p_t position
v_t velocity

p^{des} desired position

Obstacles

Obstacle i is a box $[\underline{o}^i, \overline{o}^i]$

Motion planning formulation

minimize $$\|p_T - p^{\mathrm{des}}\|_2^2 + \sum_{t=0}^{T-1} \|p_t - p^{\mathrm{des}}\|_2^2 + \gamma \|u_t\|_2^2$$

Motion planning formulation

minimize $\quad \|p_T - p^{\text{des}}\|_2^2 + \sum_{t=0}^{T-1} \|p_t - p^{\text{des}}\|_2^2 + \gamma \|u_t\|_2^2$

subject to
$$(p_{t+1}, v_{t+1}) = A(p_t, v_t) + Bu_t$$
$$p_0 = p^{\text{init}}, \quad v_0 = v^{\text{init}}$$

Dynamics

Motion planning formulation

minimize $\|p_T - p^{\mathrm{des}}\|_2^2 + \sum_{t=0}^{T-1} \|p_t - p^{\mathrm{des}}\|_2^2 + \gamma \|u_t\|_2^2$

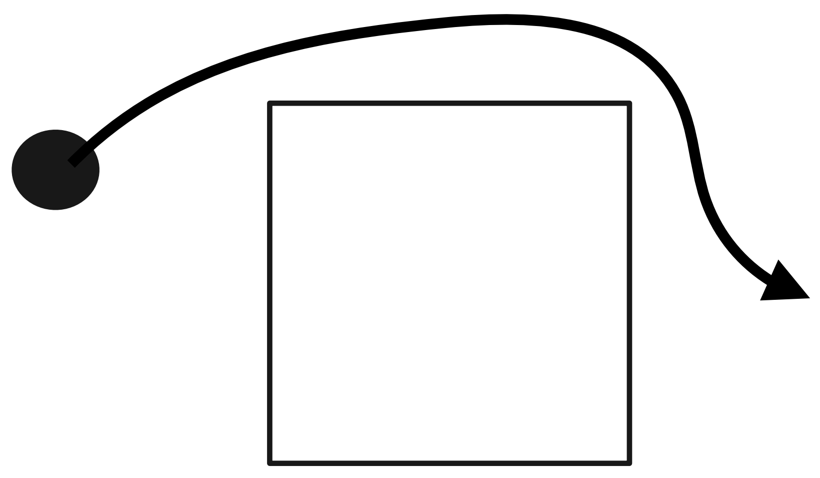
subject to

$$(p_{t+1}, v_{t+1}) = A(p_t, v_t) + Bu_t$$
$$p_0 = p^{\mathrm{init}}, \quad v_0 = v^{\mathrm{init}}$$

Dynamics

$$\overline{o}^i - M\overline{\delta}_t^i \leq p_t \leq \underline{o}^i + M\underline{\delta}_t^i, \quad i = 1, \dots, n_{\mathrm{obs}}$$
$$\mathbf{1}^T\underline{\delta}_t^i + \mathbf{1}^T\overline{\delta}_t^i \leq 2d - 1$$

Obstacle avoidance

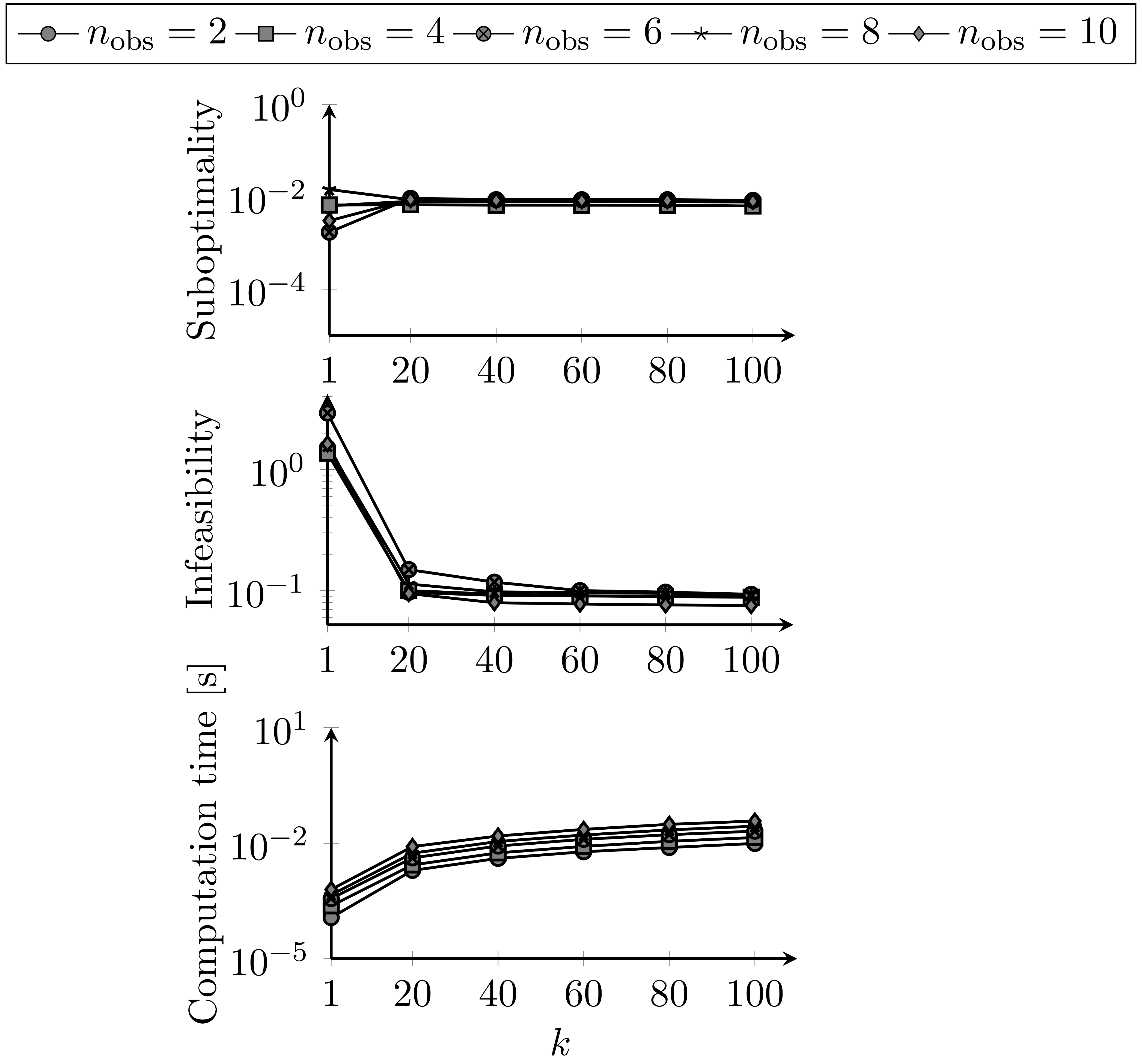


$$\overline{\delta}_t^i, \underline{\delta}_t^i \in \{0, 1\}^d, \qquad i = 1, \dots, n_{\mathrm{obs}}$$

Motion planning with obstacles

Performance

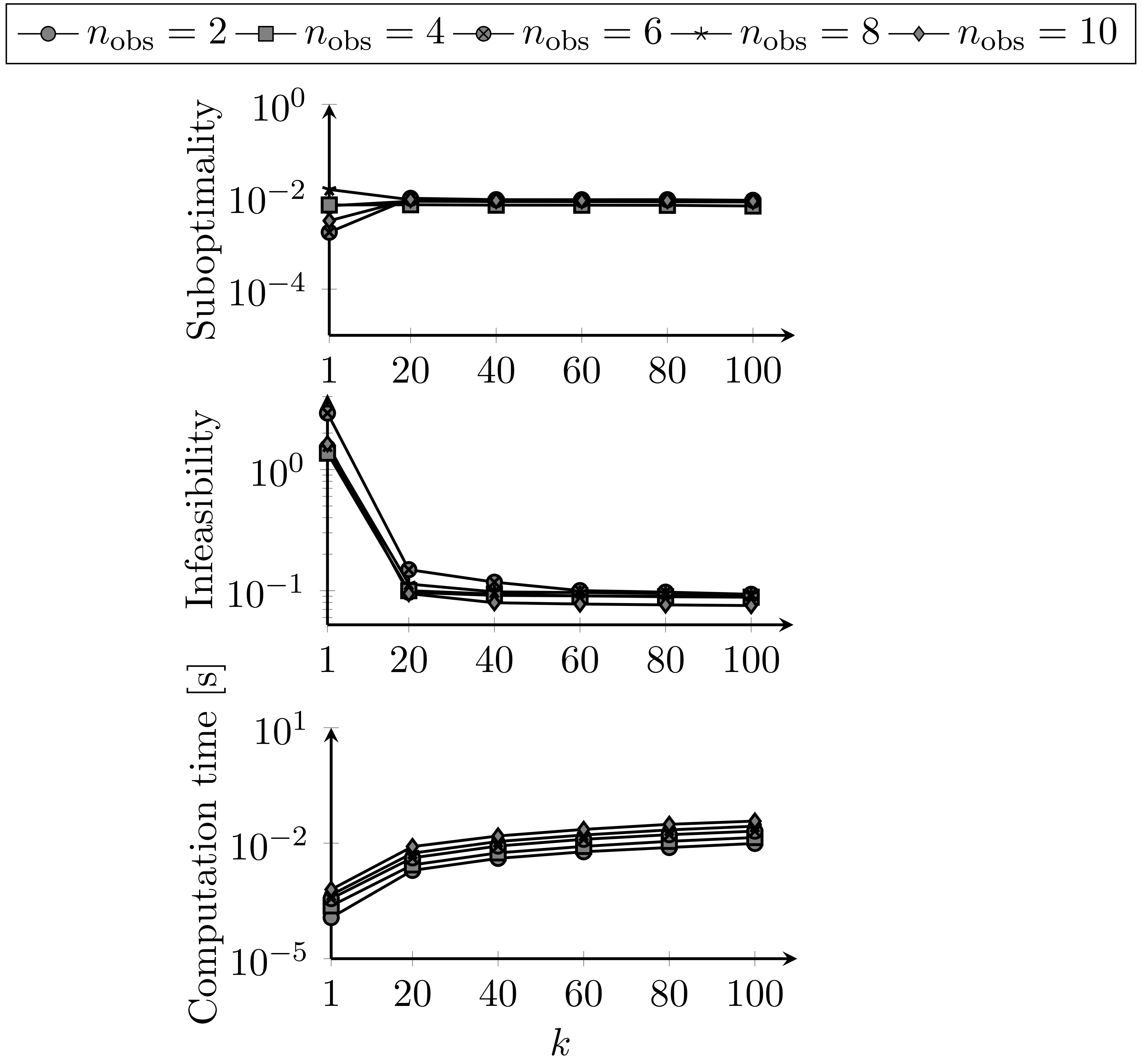
Varying top-k



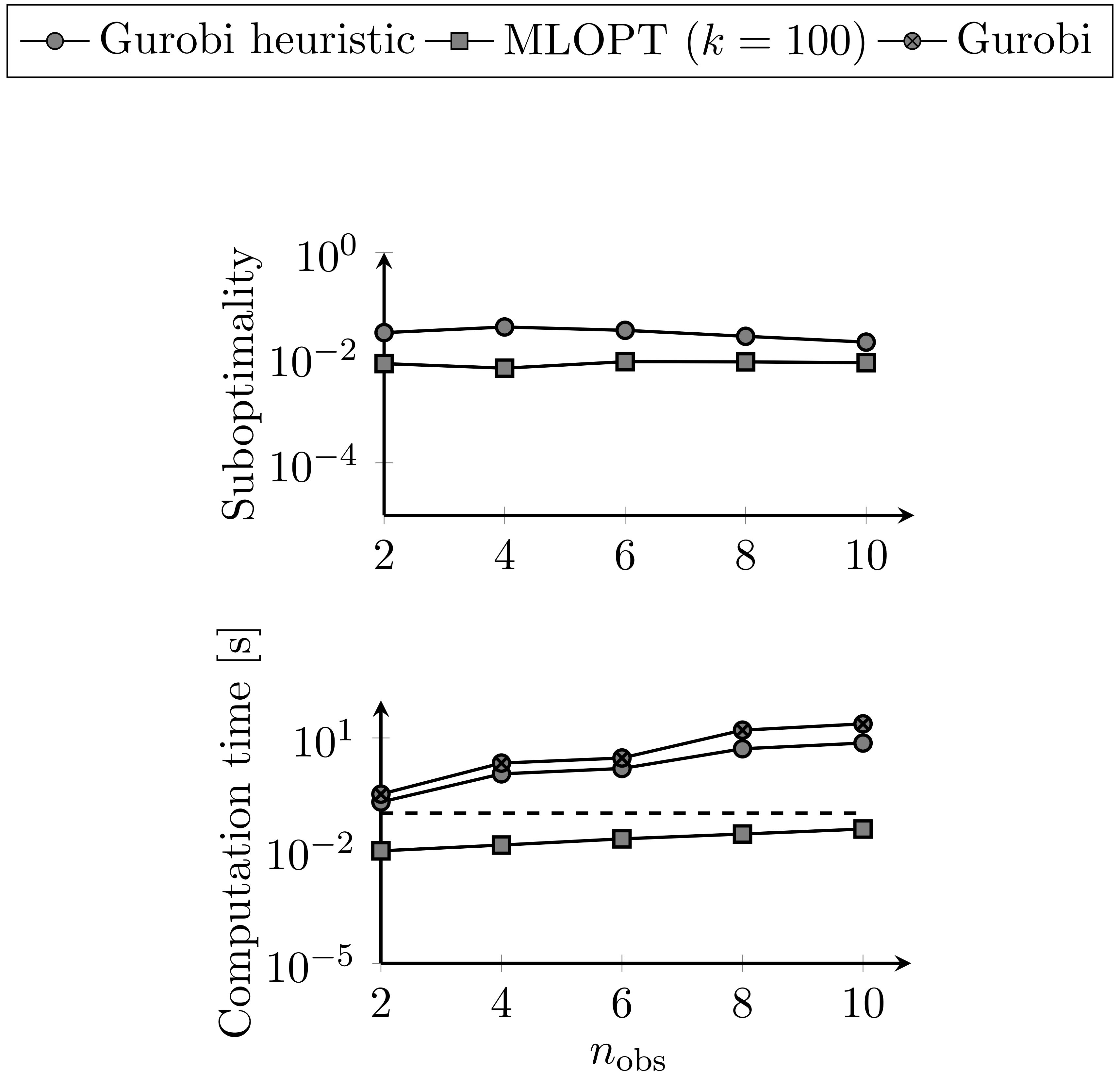
Motion planning with obstacles

Performance

Varying top-k



Varying n_{obs}



Motion planning with obstacles

Worst-case timings

$n_{\text{obstacles}}$	n_{var}	n_{constr}	M	$t_{\max}$ MLOPT [s]	$t_{\max}$ Gurobi [s]	$t_{\max}$ Gurobi heuristic [s]
2	1135	3773	1371	0.4145	2.3776	2.2962
4	1615	10133	1135	0.1878	11.8172	8.1443
6	2095	20333	939	0.3173	33.7869	11.5292
8	2575	34373	845	0.2235	392.3073	128.4948
10	3055	52253	696	0.2896	773.1476	206.4520

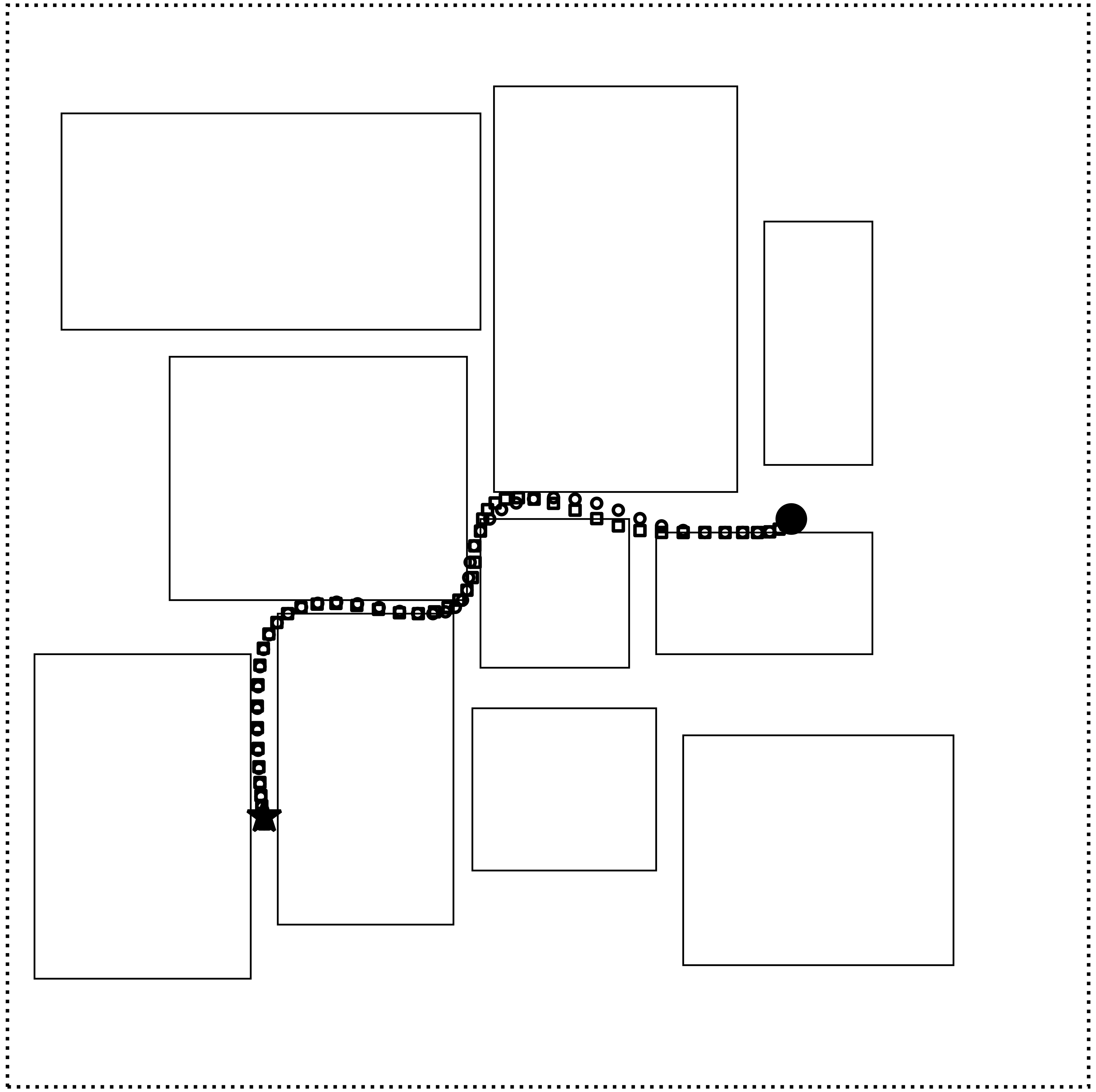
2600x speedups

Moderate number of strategies

Motion planning with obstacles

Circles
optimal

Squares
MLOPT



Conclusions

A very fast machine learning approach to online optimization

We solve online MIO in milliseconds

Machine learning classifier simplifies nonconvexities

We prune redundant labels

Conclusions

A very fast machine learning approach to online optimization

We solve online MIO in milliseconds

Machine learning classifier simplifies nonconvexities

We prune redundant labels

Refrences

[D. Bertsimas, B. Stellato, The Voice of Optimization, Machine Learning, (2020)]

[D. Bertsimas, B. Stellato, *Online Mixed-Integer Optimization in Milliseconds*, arXiv 1907.02206, (2020)]

[A. Cauligi, P. Culbertson, B. Stellato, D. Bertsimas, M. Schwager, M. Pavone, *Learning Mixed-Integer Convex Optimization Strategies for Robot Planning and Control*, IEEE Conference on Decision and Control (2020)]

Conclusions

A very fast machine learning approach to online optimization

We solve online MIO in milliseconds

Machine learning classifier simplifies nonconvexities

We prune redundant labels

Refrences

[D. Bertsimas, B. Stellato, The Voice of Optimization, Machine Learning, (2020)]

[D. Bertsimas, B. Stellato, *Online Mixed-Integer Optimization in Milliseconds*, arXiv 1907.02206, (2020)]

[A. Cauligi, P. Culbertson, B. Stellato, D. Bertsimas, M. Schwager, M. Pavone, *Learning Mixed-Integer Convex Optimization Strategies for Robot Planning and Control*, IEEE Conference on Decision and Control (2020)]

stellato.io

bstellato@princeton.edu

@b_stellato

github.com/bstellato/mlopt