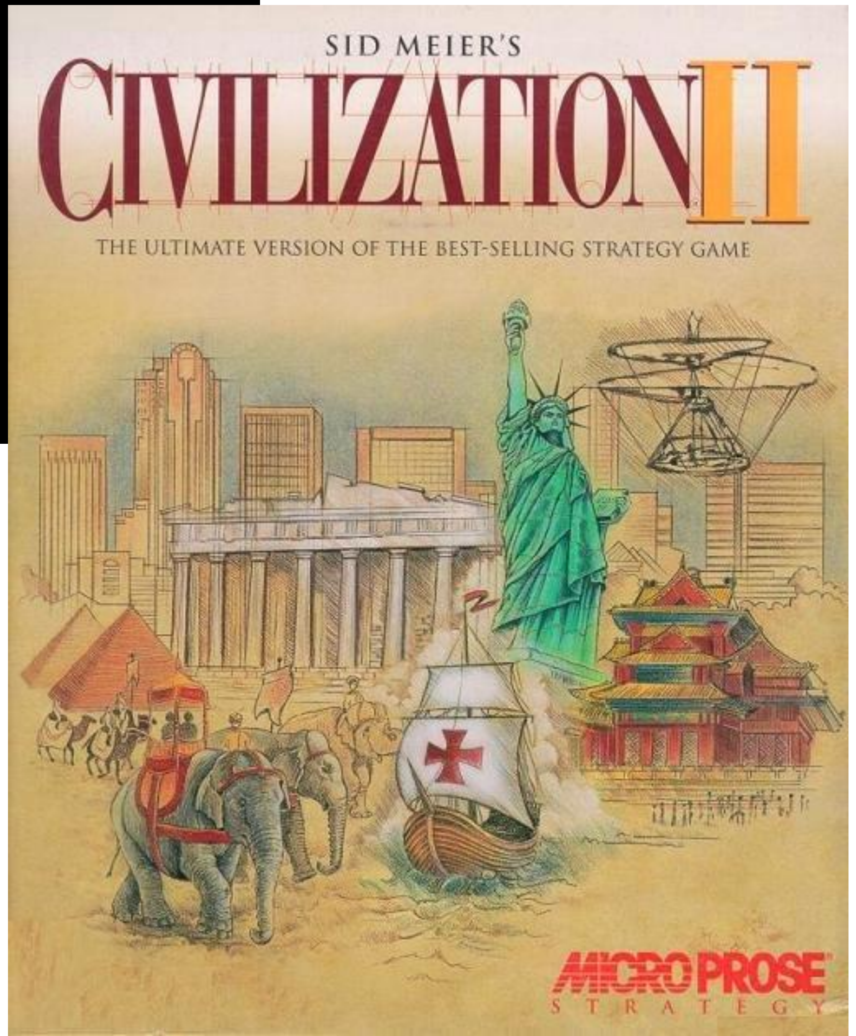
Monte Carlo Tree Search in **continuous spaces** using Voronoi optimistic optimization with **regret bounds**

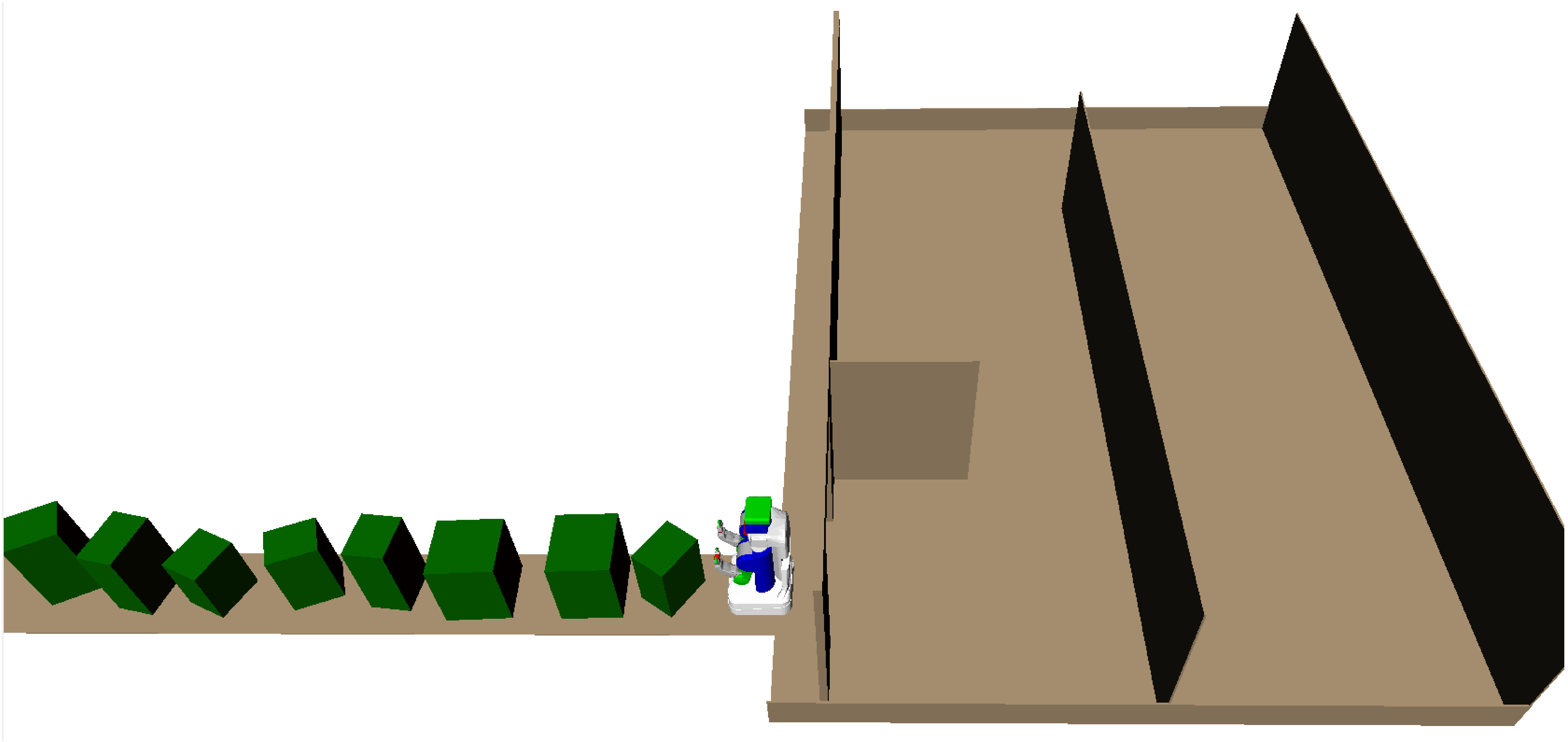
KAIST AI
Graduate School of AI

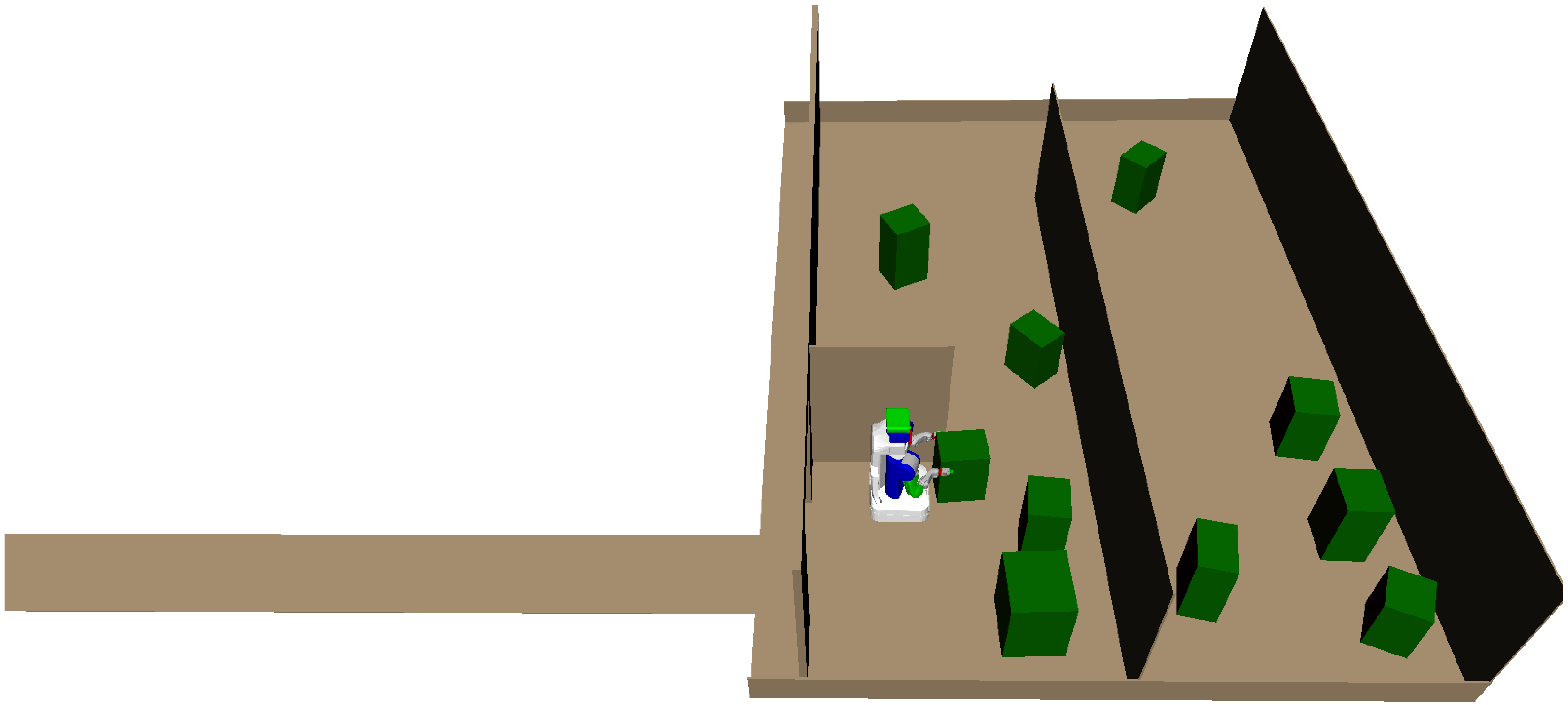
Beomjoon Kim

(with Kyungjae Lee, Sungbin Lim, Leslie Pack Kaelbling, Tomas Lozano-Perez)

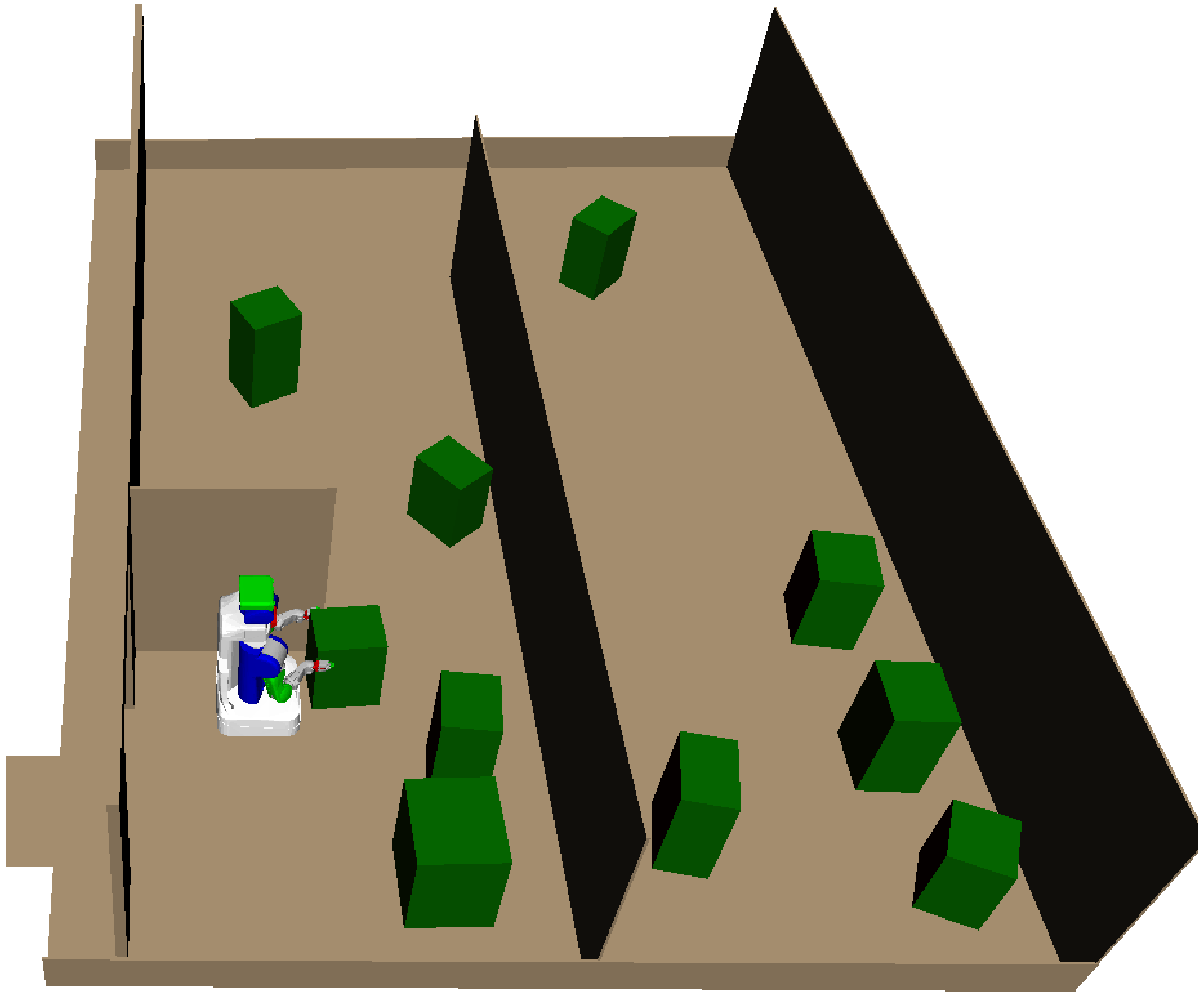
Monte Carlo Tree Search in **discrete** action spaces



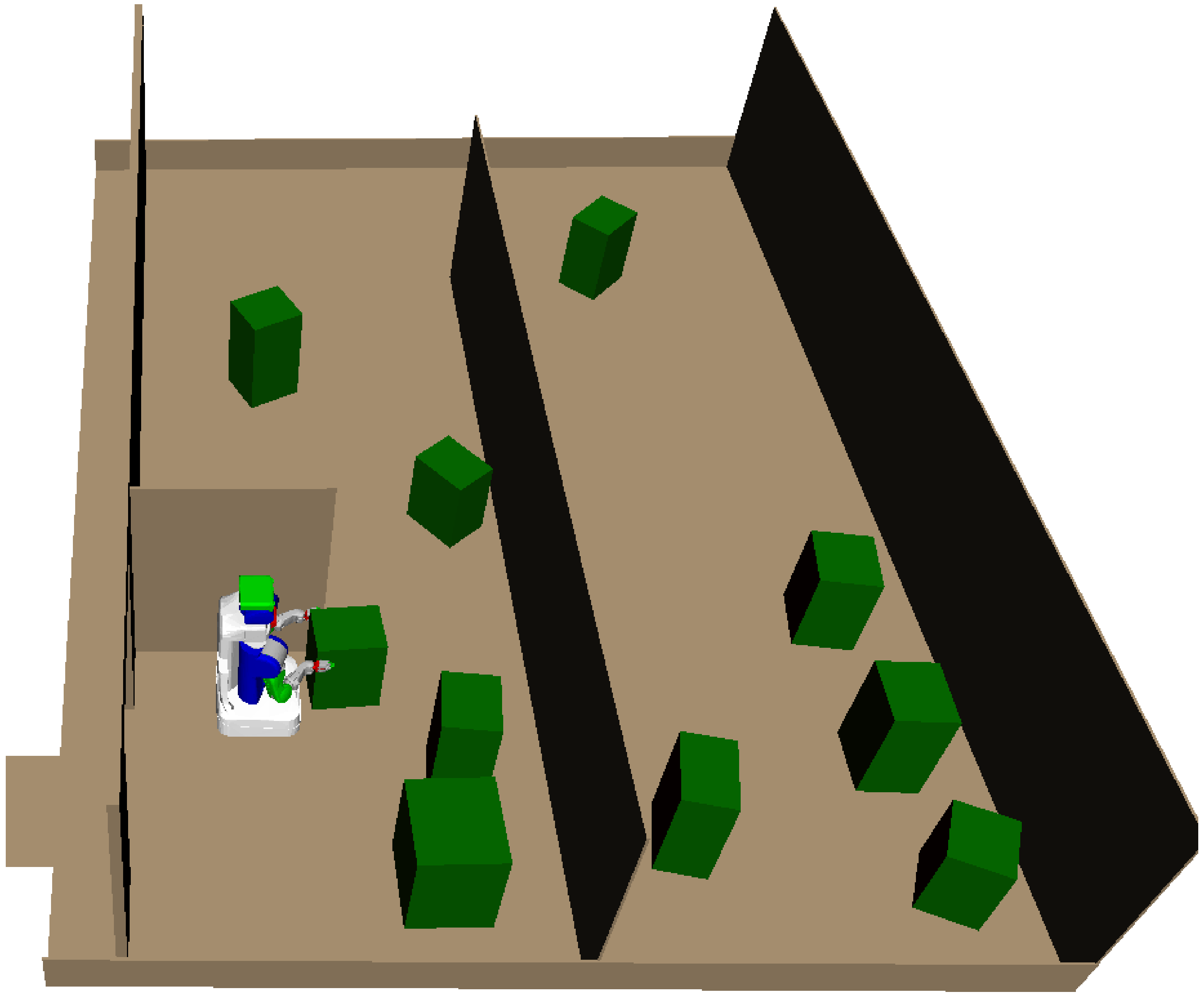




Characteristics of **robotics** problems



Characteristics of **robotics** problems



Continuous and high-dimensional action spaces

Main question addressed in our paper

How can we extend MCTS to **high-dimensional continuous action space** problems?

First, consider the easier problem

- Assume **planning horizon is 1**

First, consider the easier problem

- Assume **planning horizon is 1**
- The problem is reduced to a **budgeted-black-box-function optimization** (BBFO) problem

First, consider the easier problem

- Assume **planning horizon is 1**
- The problem is reduced to a **budgeted-black-box-function optimization** (BBFO) problem

$$\max_{a \in \mathcal{A}} R(a)$$

First, consider the easier problem

- Assume **planning horizon is 1**
- The problem is reduced to a **budgeted-black-box-function optimization** (BBFO) problem

$\max_{a \in \mathcal{A}} R(a)$ within **n number of evaluations**

BBFO: general problem formulation

BBFO: general problem formulation

- Given:
  - Budget n
  - Bounded search space $\mathcal{X}$
  - Objective function f

BBFO problem formulation

- Goal: minimize the ***simple regret***

$$f(x_\star) - \max_{t \in [n]} f(x_t)$$

A class of BBFO methods

Bayesian Optimization (BO)

EI (Mockus, 1974)
PI (Kushner, 1964)
GP-UCB (Srinivas et al., 2010)
ES (Hennig & Schuler, 2012)
PES (Hernandez-Lobato et al., 2014)
EST (Wang et al., 2016)

Drawback in high-dimensional search spaces

Bayesian Optimization (BO)

Requires global optimization of the acquisition function

- EI (Mockus, 1974)
- PI (Kushner, 1964)
- GP-UCB(Auer, 2002; Srinivas et al., 2010)
- ES (Hennig & Schuler, 2012)
- PES (Hernandez-Lobato et al., 2014)
- EST (Wang et al., 2016)

Another class of BBFO methods

Bayesian Optimization (BO)

Requires global optimization of the acquisition function

- EI (Mockus, 1974)
- PI (Kushner, 1964)
- GP-UCB(Auer, 2002; Srinivas et al., 2010)
- ES (Hennig & Schuler, 2012)
- PES (Hernandez-Lobato et al., 2014)
- EST (Wang et al., 2016)

Hierarchical Partitioning (HP)

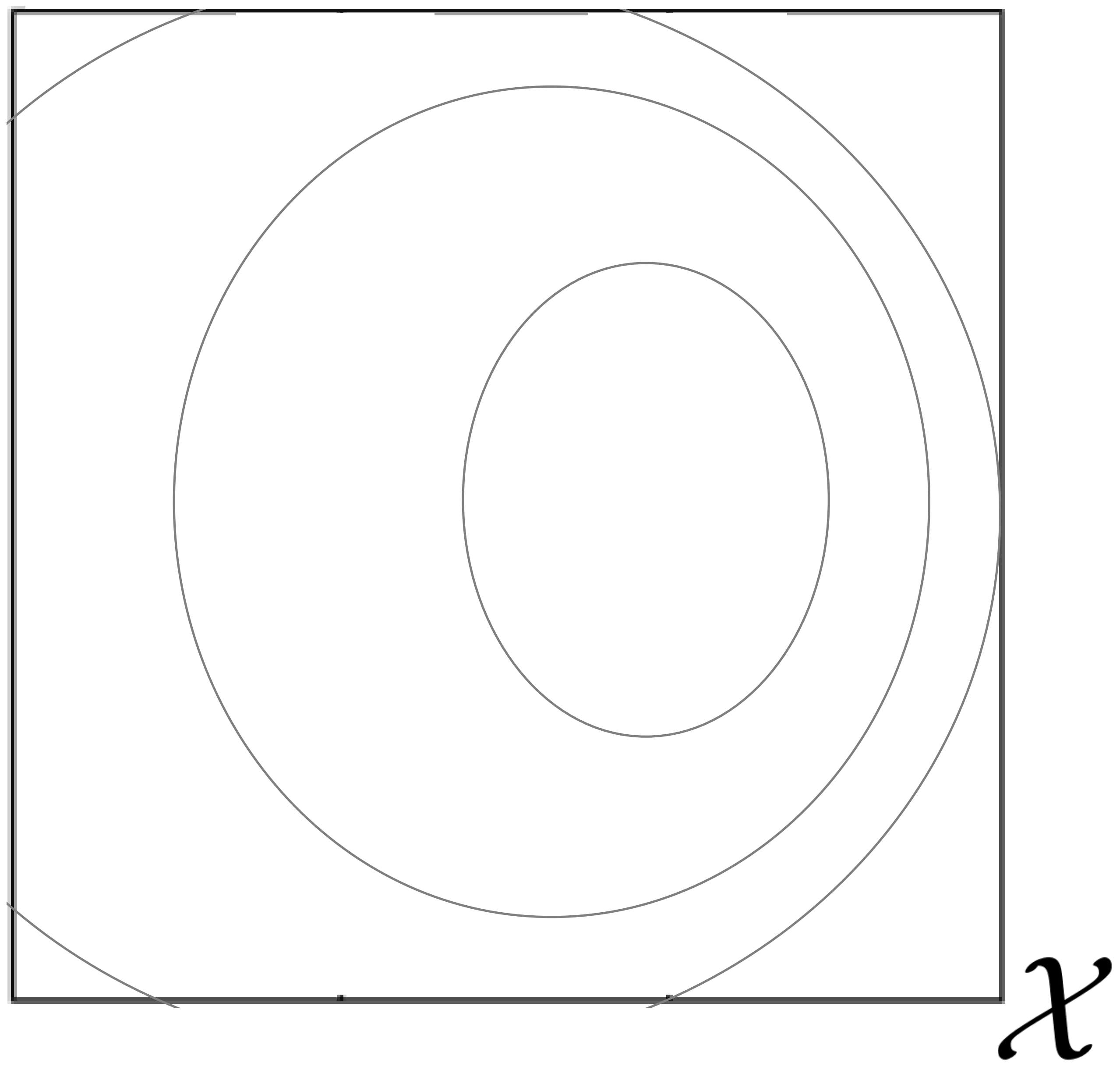
DOO (Munos, 2011)
SOO (Valko et al., 2013)
HOO (Bubeck et al., 2010)

How hierarchical partitioning methods work

- Consecutively partitioning the search space

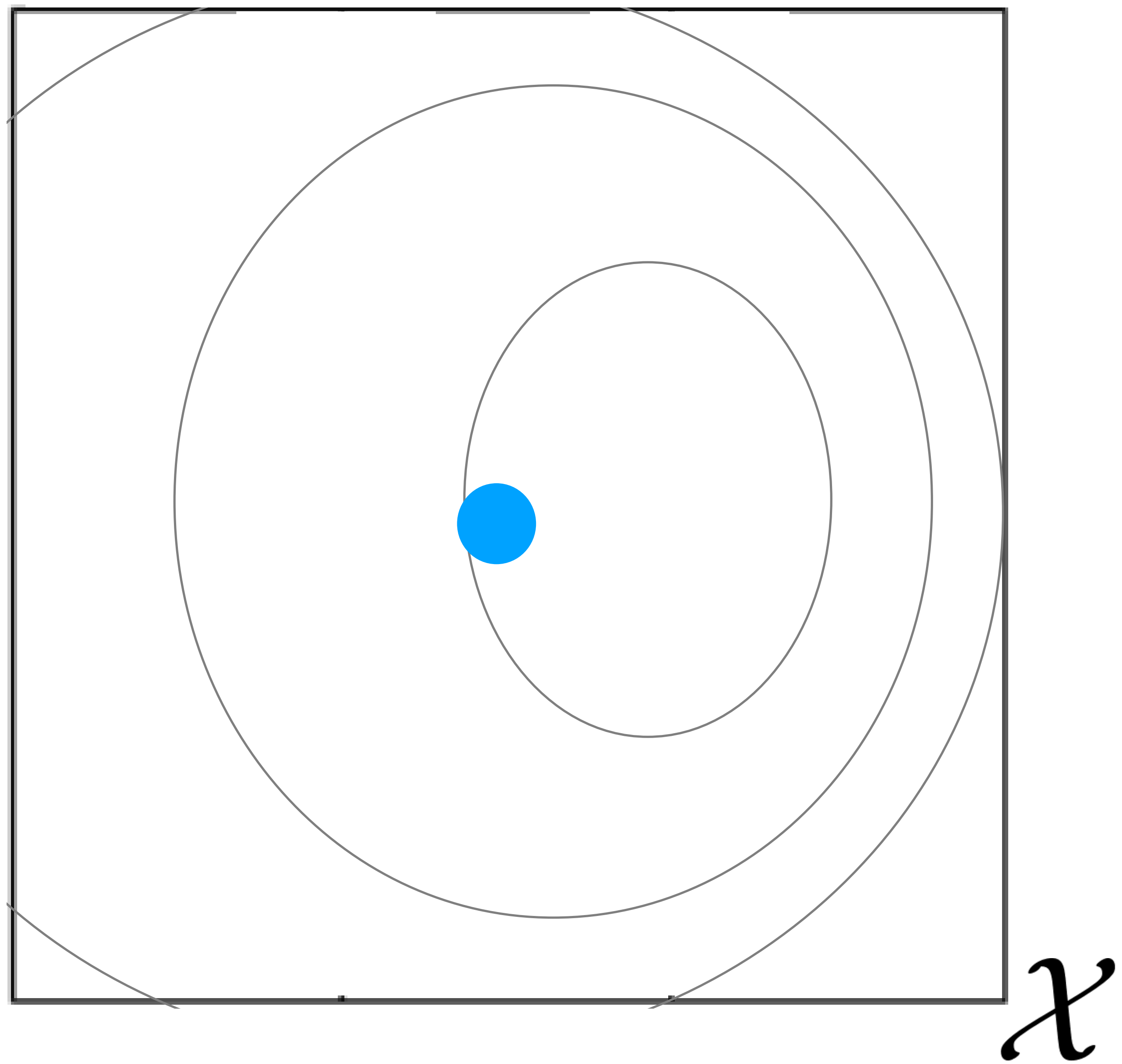
How hierarchical partitioning methods work

- Consecutively partitioning the search space



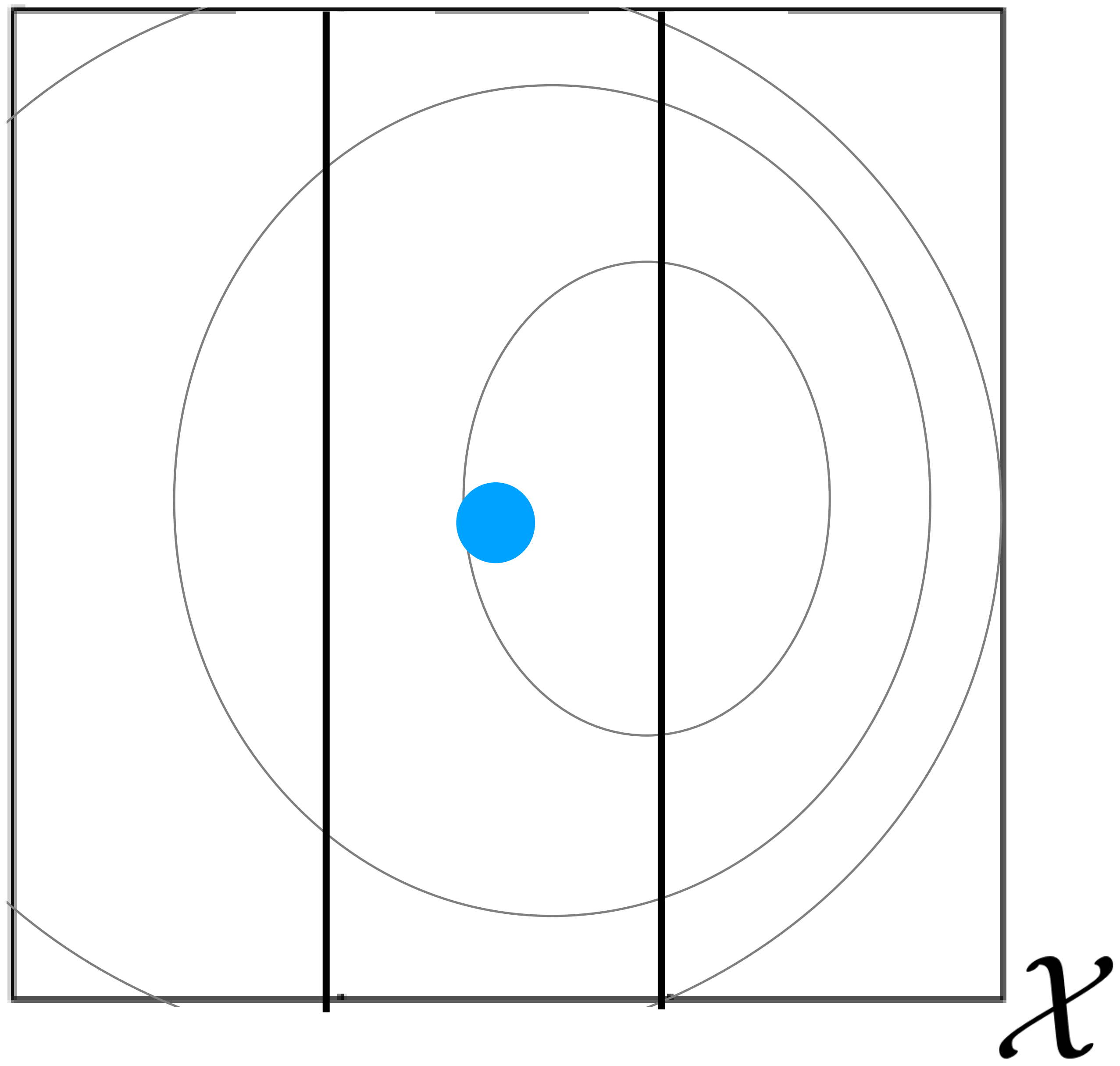
How hierarchical partitioning methods work

- First evaluated point



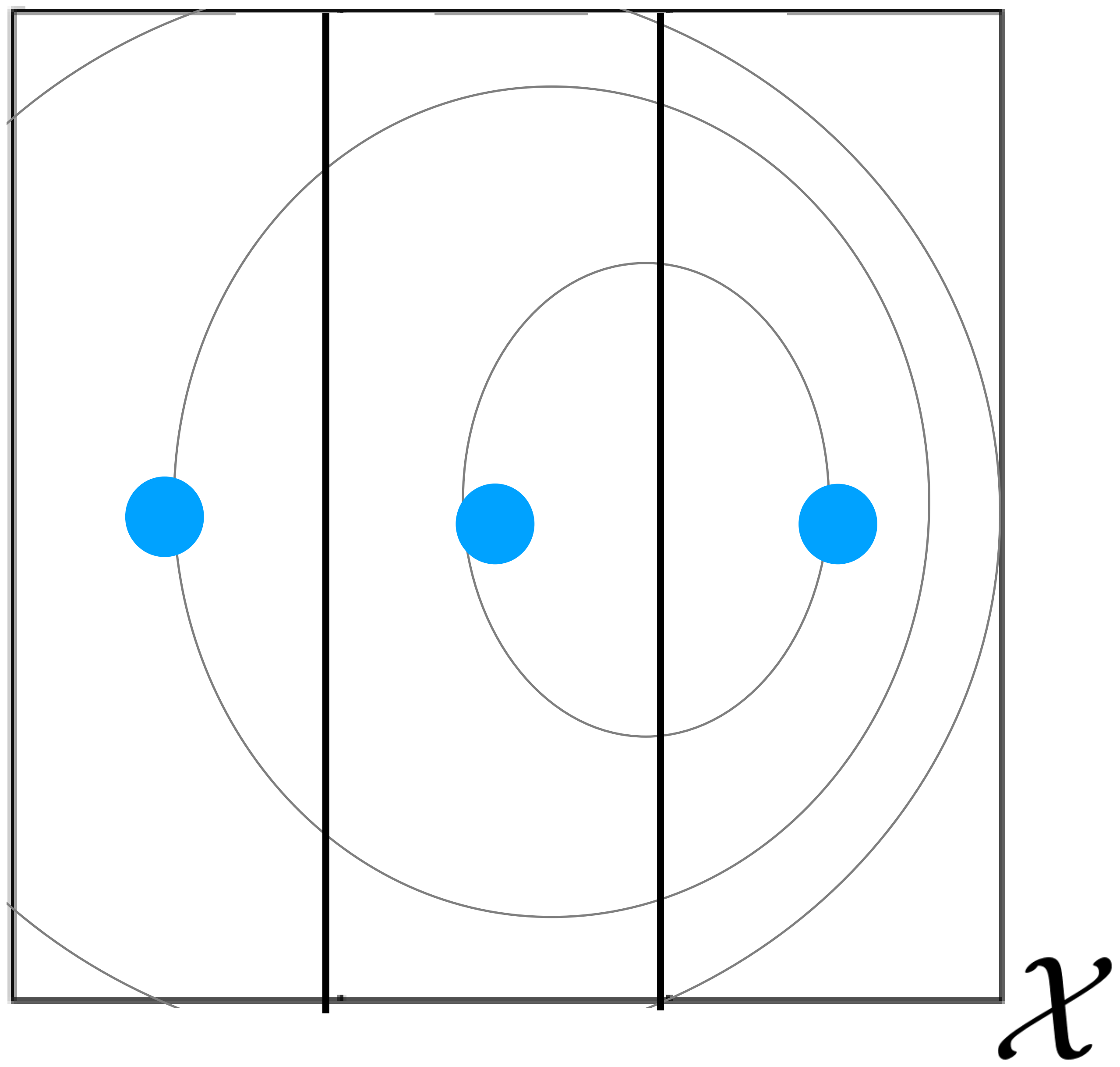
How hierarchical partitioning methods work

- Partition space by constructing K cells. Let's say K = 3



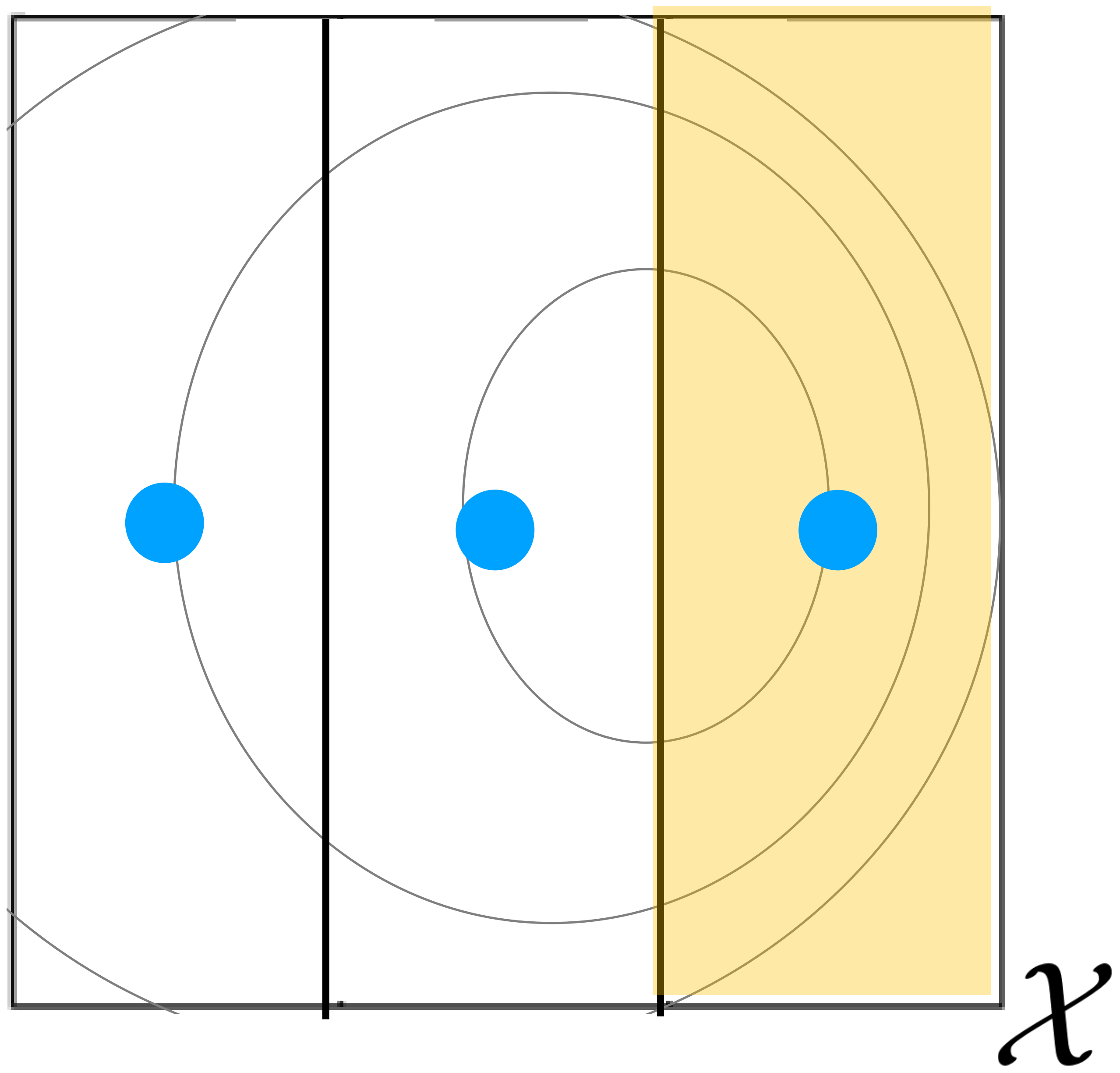
How hierarchical partitioning methods work

- Evaluate points in the new cells



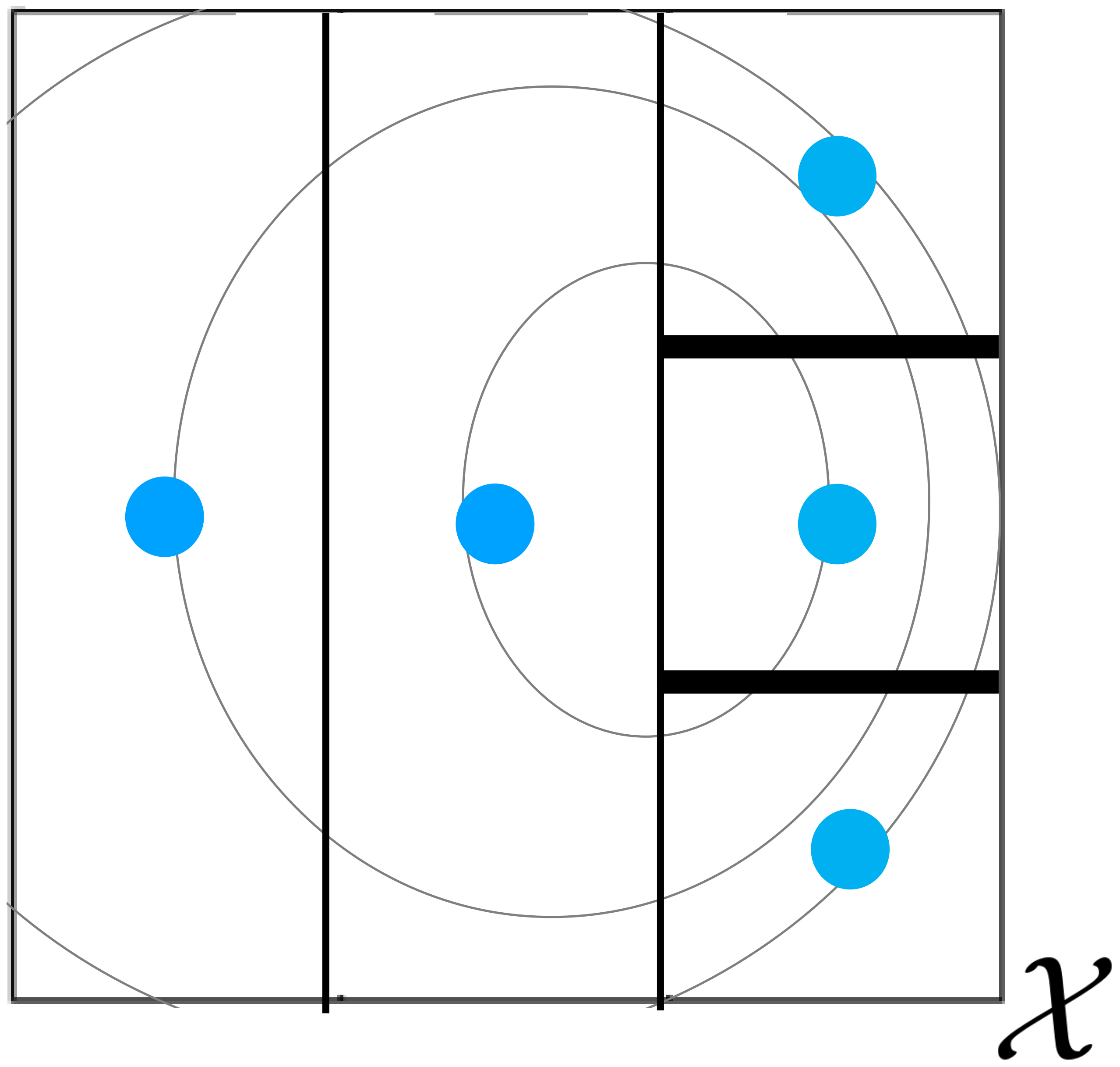
How hierarchical partitioning methods work

- Choose the most promising cell



How hierarchical partitioning methods work

- Construct a new partition, and evaluate points



How to select the **most promising cell?**

Selecting the **most promising cell**

- **Deterministic Optimistic Optimization** [Munos 2011]

Selecting the **most promising cell**

- **Deterministic Optimistic Optimization** [Munos 2011]
- Compute the b-value of each cell, where

$$b(x_i)$$

Selecting the **most promising cell**

- **Deterministic Optimistic Optimization** [Munos 2011]
- Compute the b-value of each cell, where

$$b(x_i) = f(x_i)$$

Selecting the **most promising cell**

- **Deterministic Optimistic Optimization** [Munos 2011]
- Compute the b-value of each cell, where

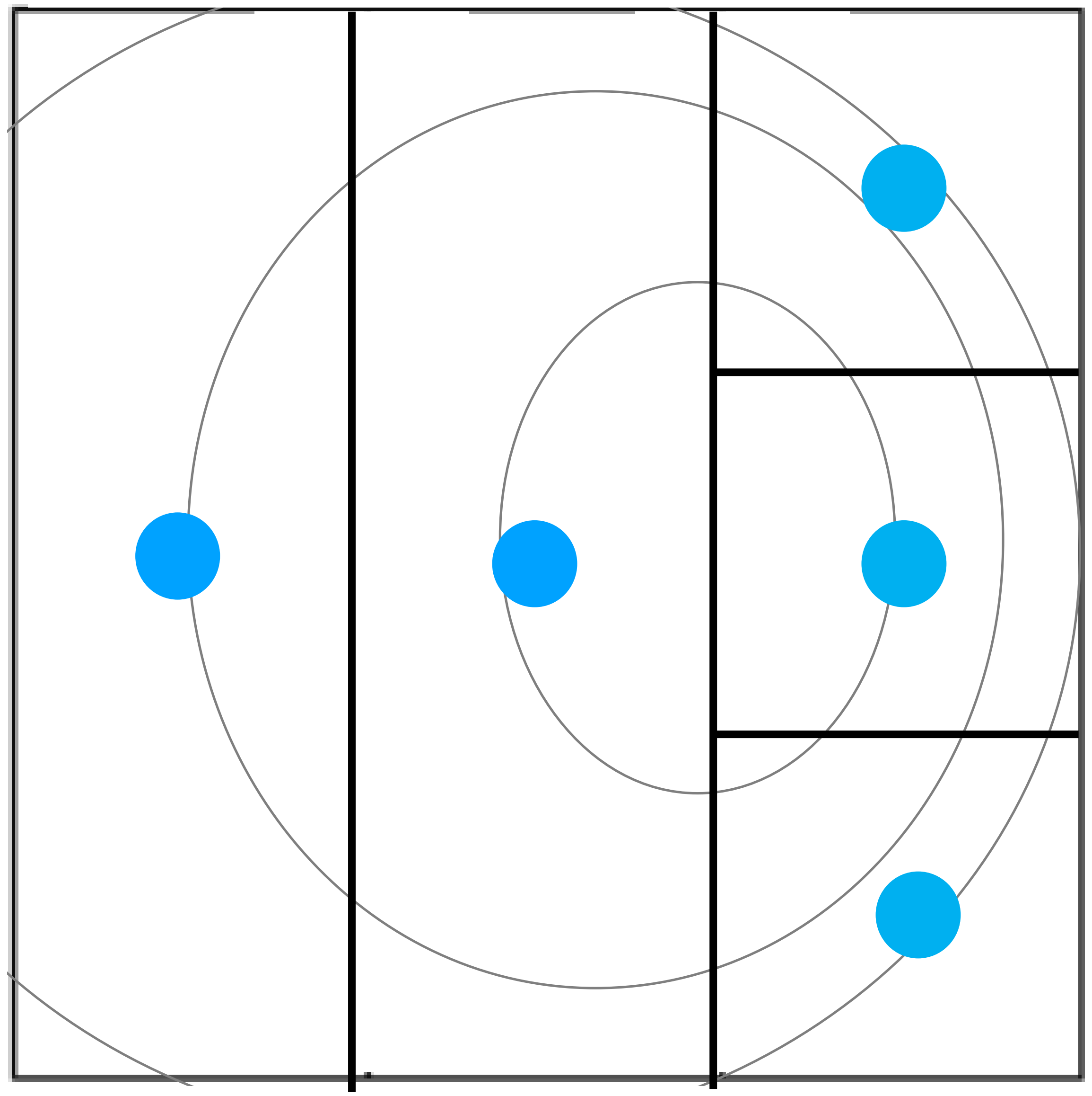
Exploration bonus

$$b(x_i) = f(x_i) + \boxed{\delta(i)}$$

Prefers larger cells

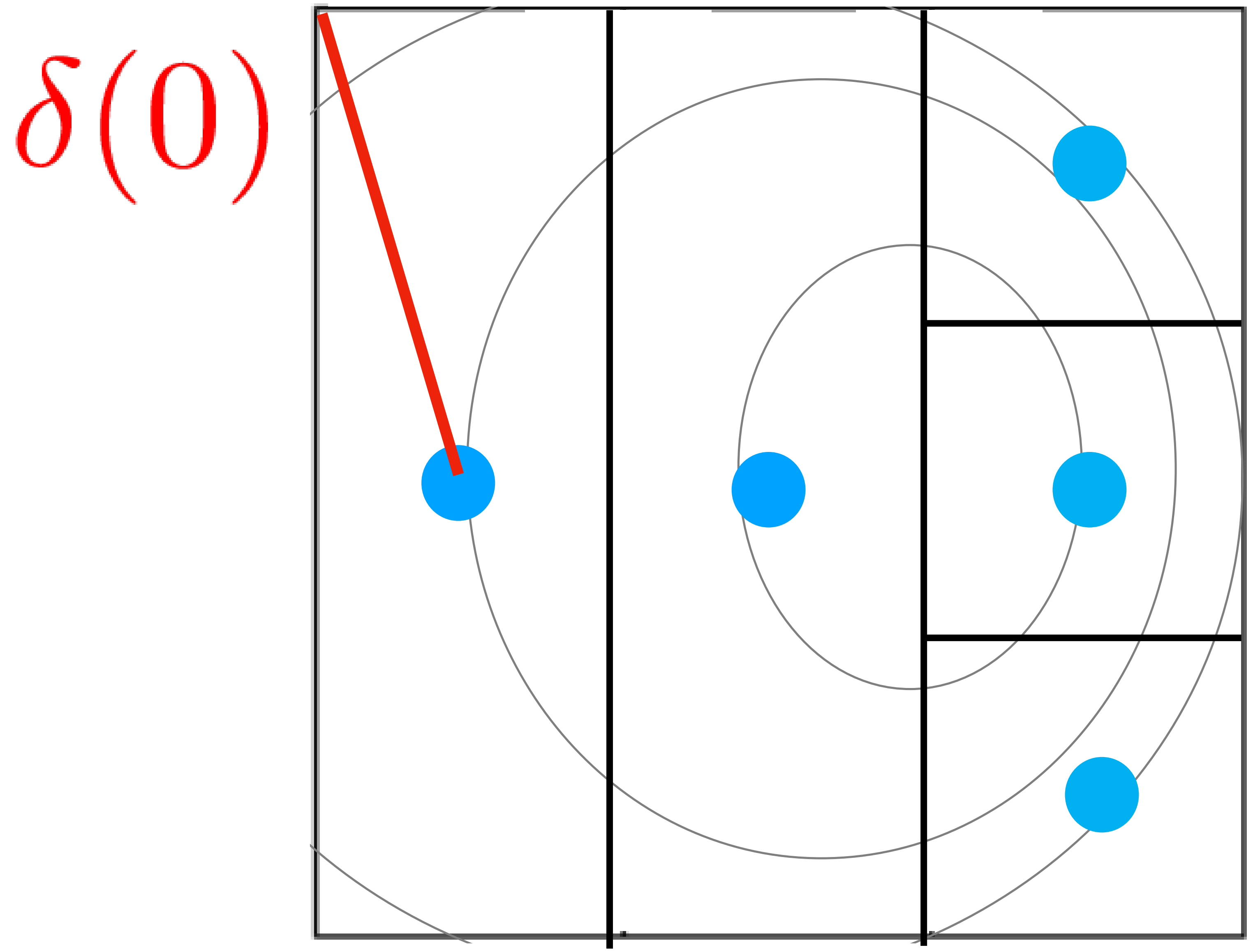
Diameter of a cell

$$b(x_i) = f(x_i) + \delta(i)$$



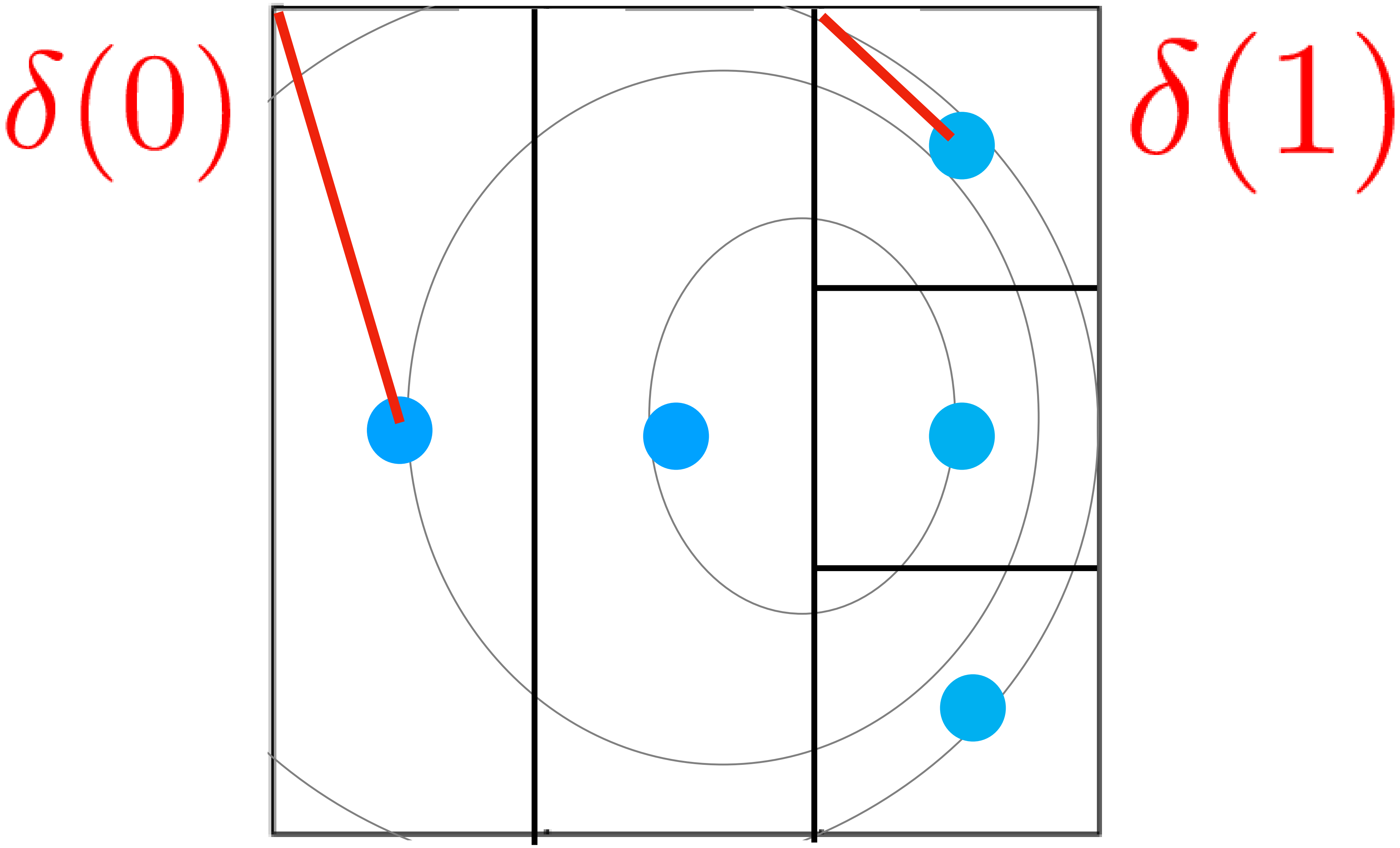
Diameter of a cell

$$b(x_i) = f(x_i) + \delta(i)$$



Diameter of a cell

$$b(x_i) = f(x_i) + \delta(i)$$



DOO: **Choose the cell with the highest b-value**

$$\arg\max_i f(x_i) + \delta(i)$$

DOO: **exploration vs. exploitation**

$$\arg\max_i f(x_i) + \delta(i)$$

DOO: **exploration vs. exploitation**

Exploitation of current function knowledge

prefer cells that have high function values

$$\arg\max_i \boxed{f(x_i)} + \delta(i)$$

DOO: **exploration vs. exploitation**

Exploitation of current function knowledge

prefer cells that have high function values

$$\arg\max_i f(x_i) + \delta(i)$$

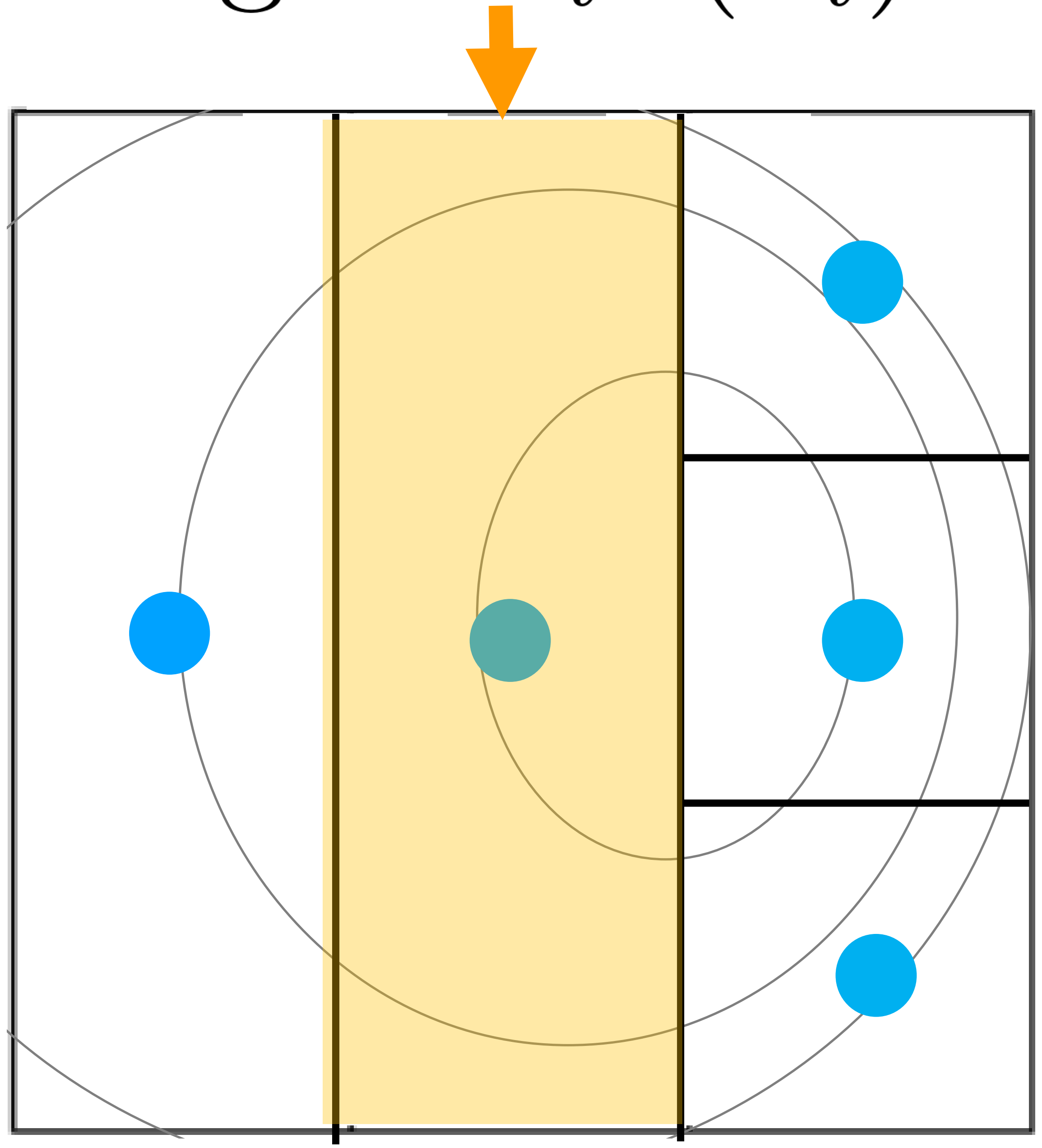
Exploration of the search space

prefer cells with large diameters

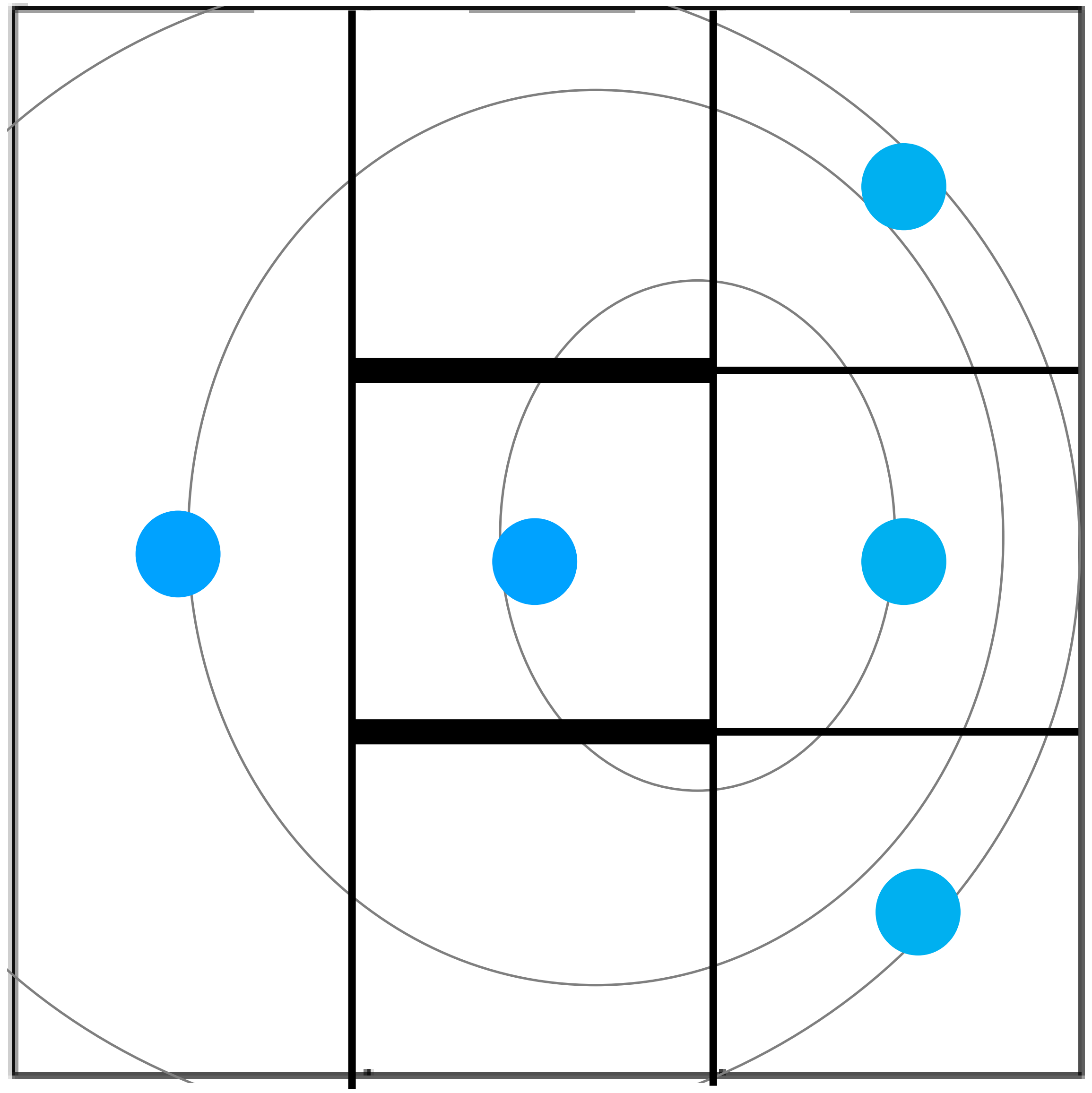
BUT: it requires **deciding which dimension to cut**

BUT: it requires **deciding which dimension to cut**

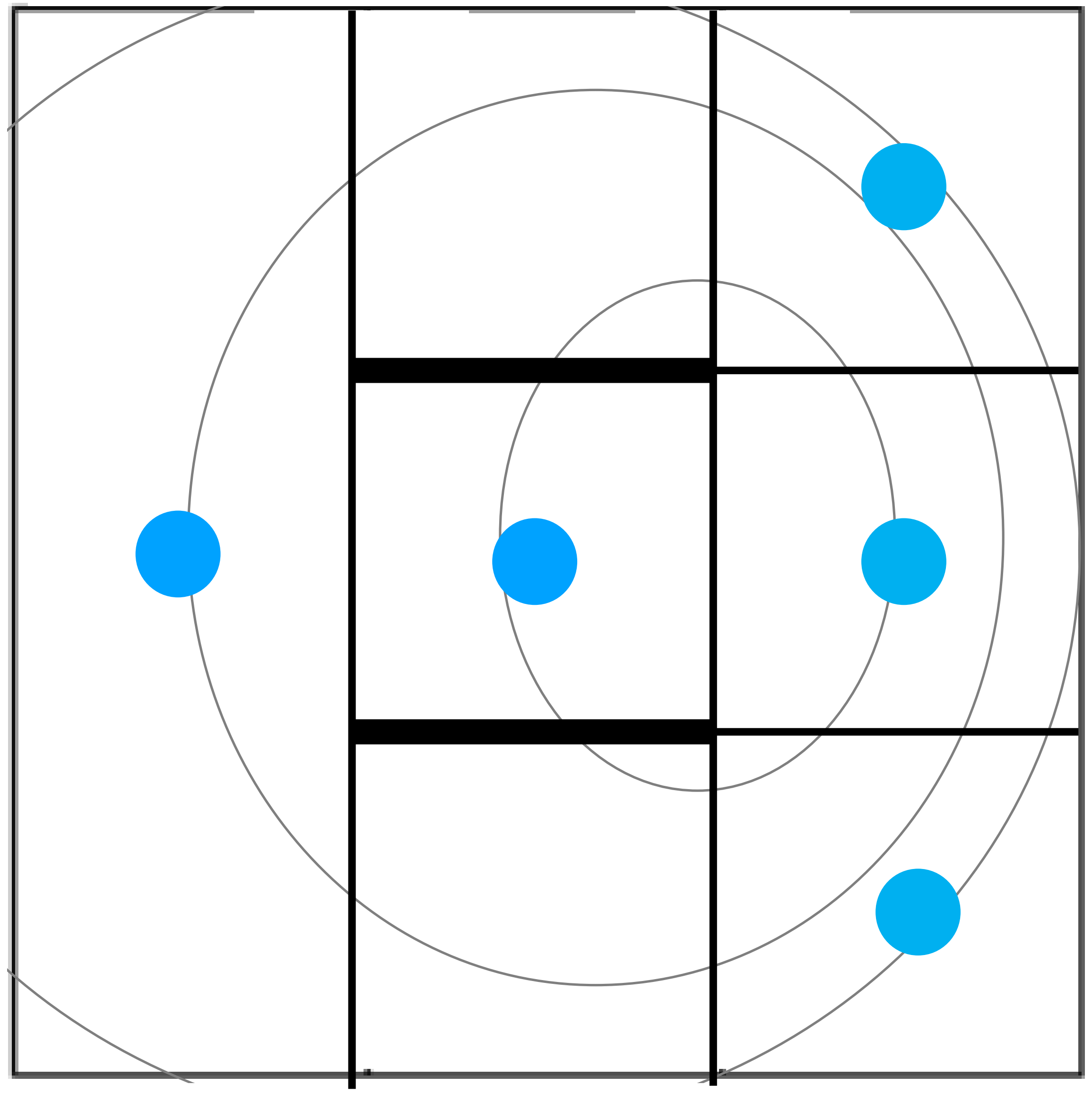
$$\arg\max_i b(x_i)$$



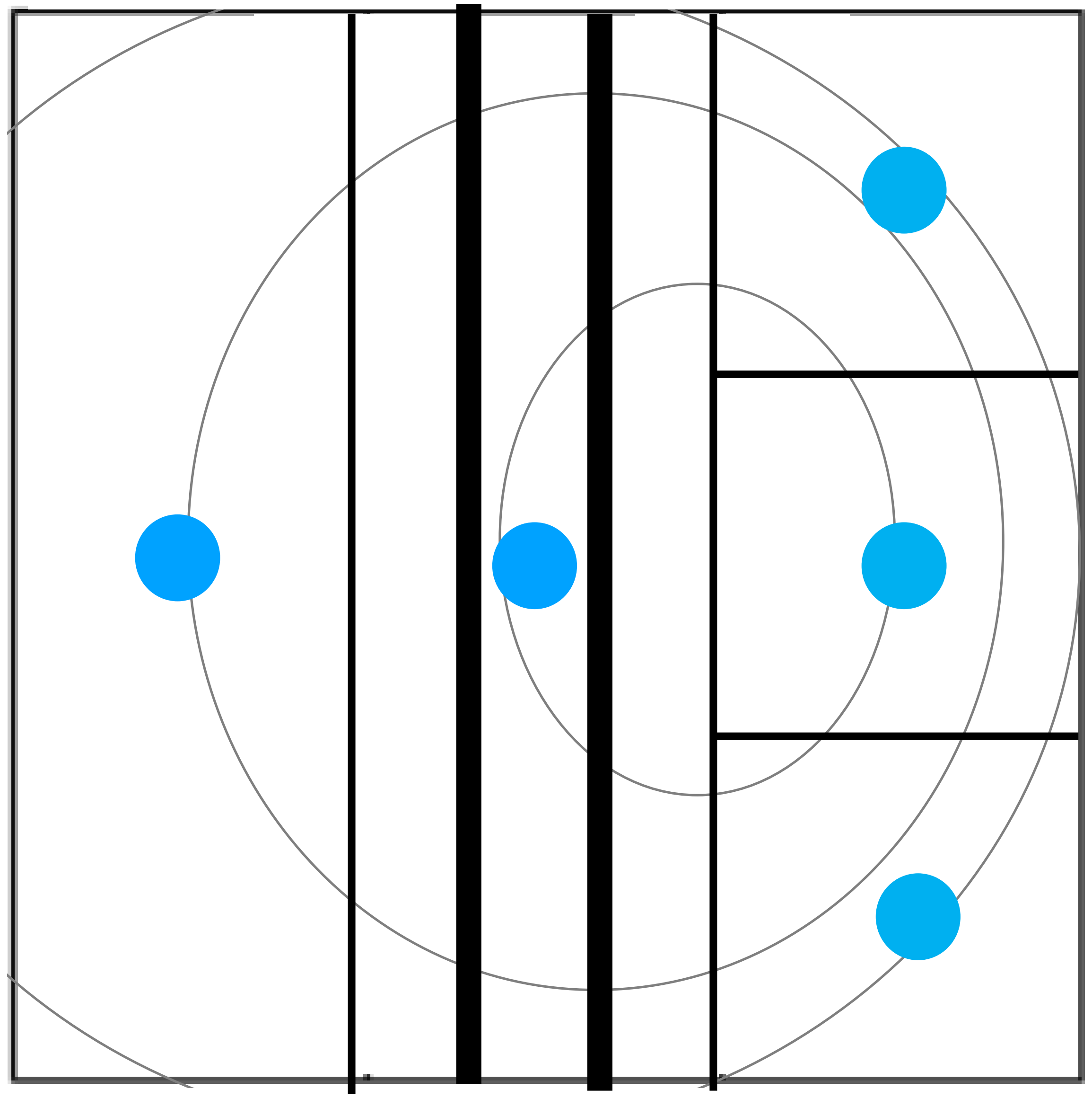
BUT: it requires **deciding which dimension to cut**



BUT: it requires **deciding which dimension to cut**



VS.



BUT: it requires **deciding which dimension to cut**

In D-dimensional search space, you have D^n number of choices

BUT: it requires **deciding which dimension to cut**

In D-dimensional search space, you have D^n number of choices

Finding the ***optimal sequence of dimensions*** to cut is not trivial

DOO's intuitive idea

- DOO idea:
  - The **larger the cell**, the more the **exploration** bonus
  - The **larger the value of the cell**, the more the **exploitation** bonus

Our main question

DOO idea:

How can we use this idea **without cell constructions**?

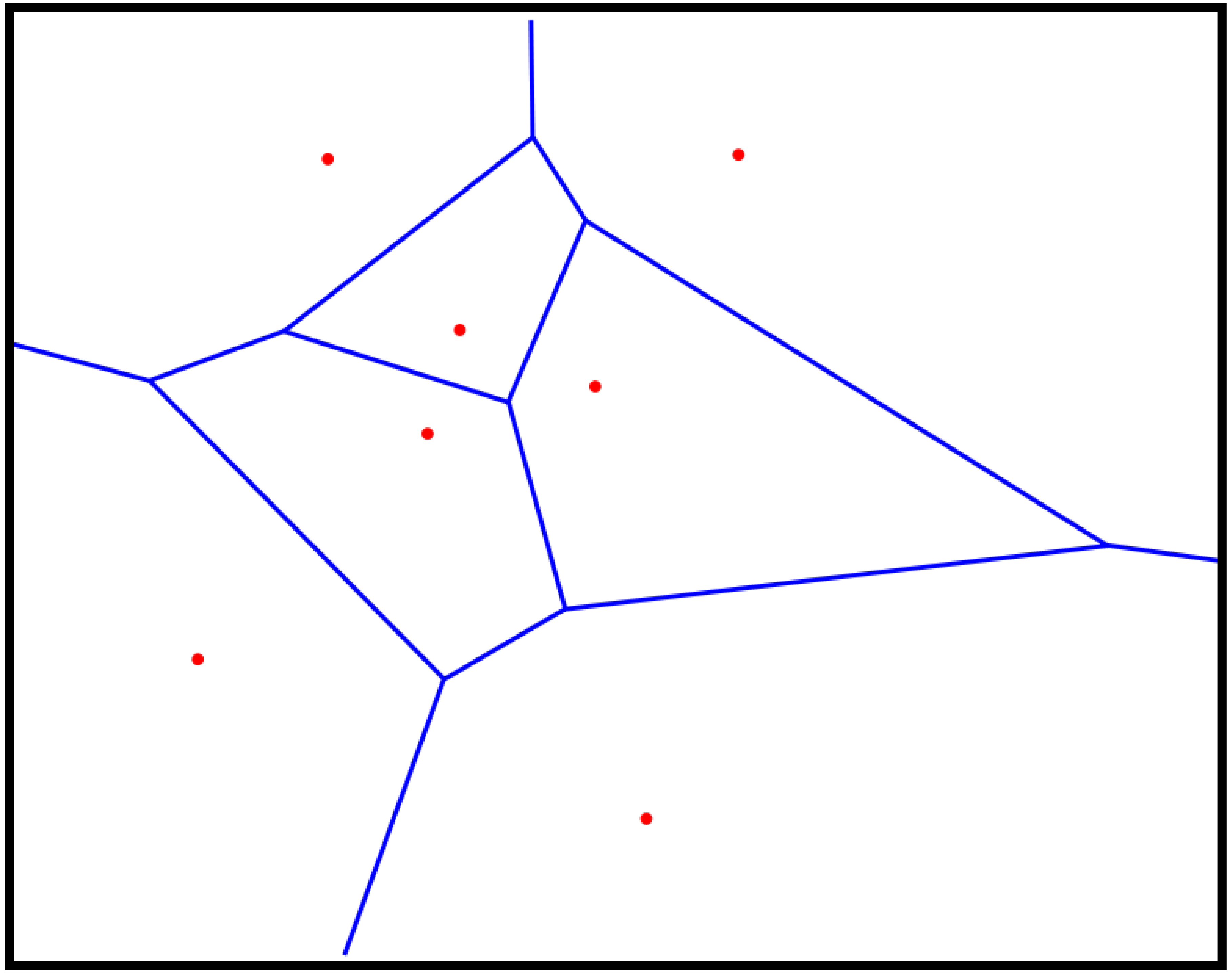
The larger the cell, the more the exploration bonus

The larger the value of the cell, the more the exploitation bonus

Our method: **Implicitly** construct **cells**

Our method: **Implicitly** construct **Voronoi cells**

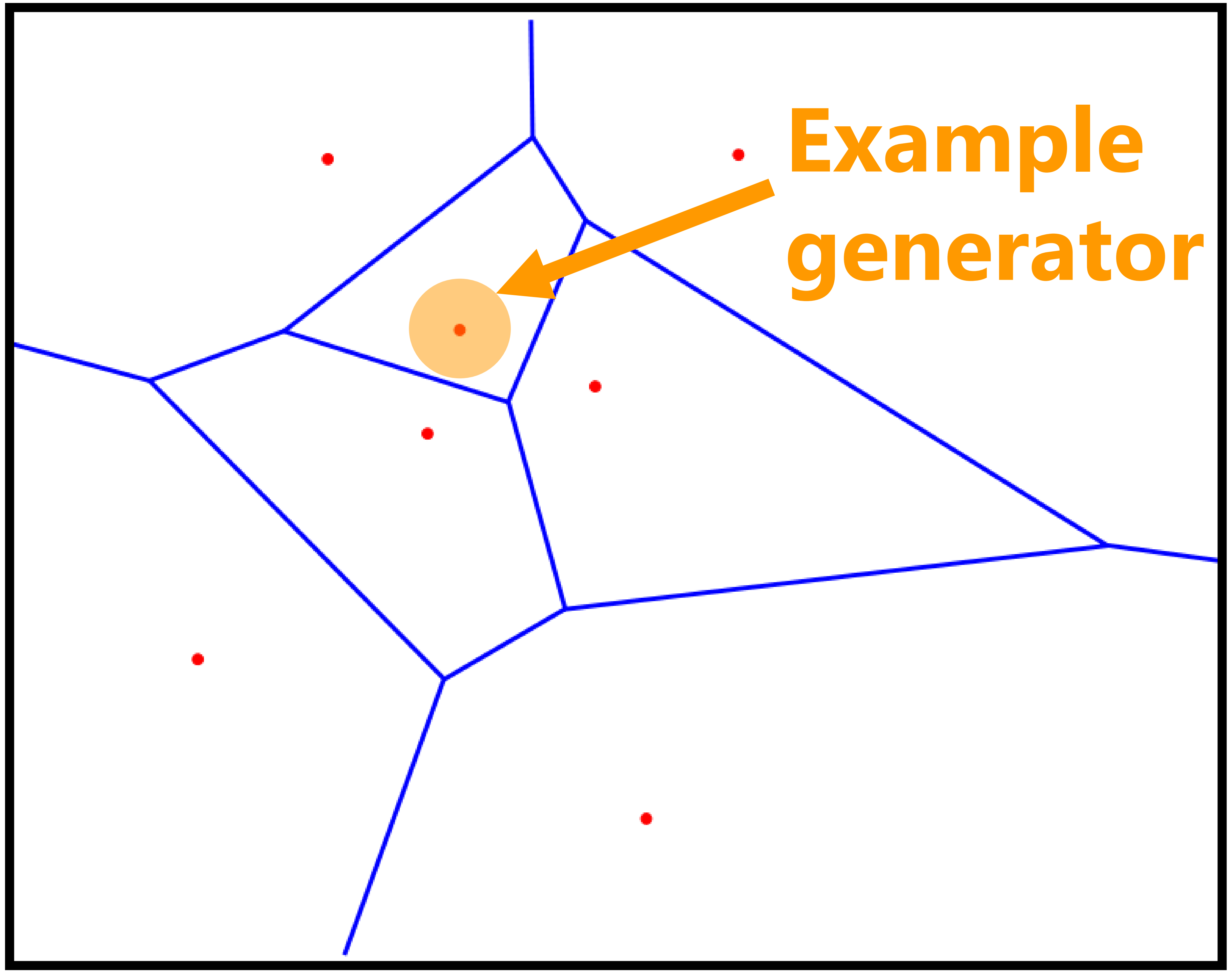
Our method: **Implicitly** construct **Voronoi cells**



Voronoi cell:

A set of points closer to its generator than all the other generators

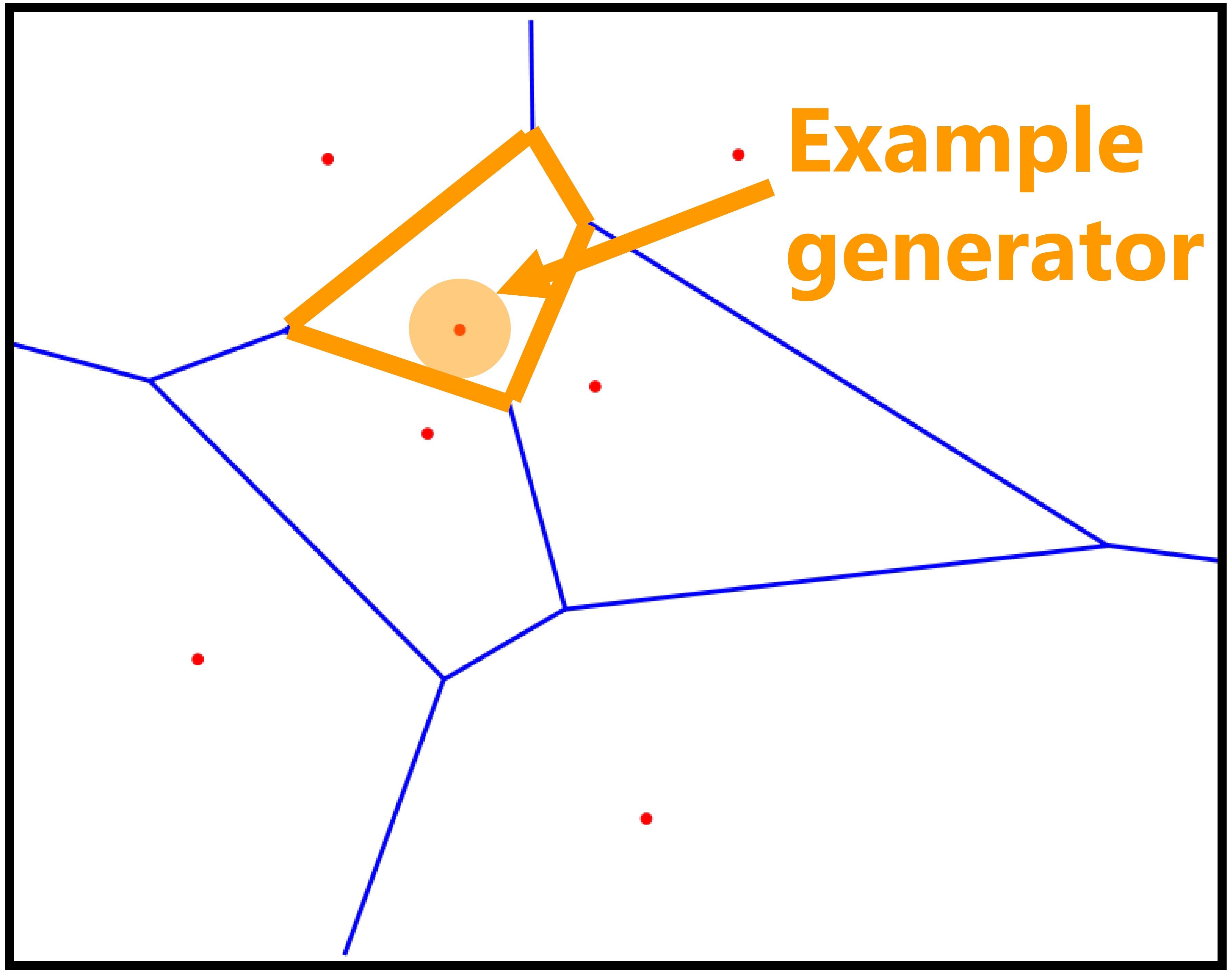
Our method: **Implicitly** construct **Voronoi cells**



Voronoi cell:

A set of points closer to its generator than all the other generators

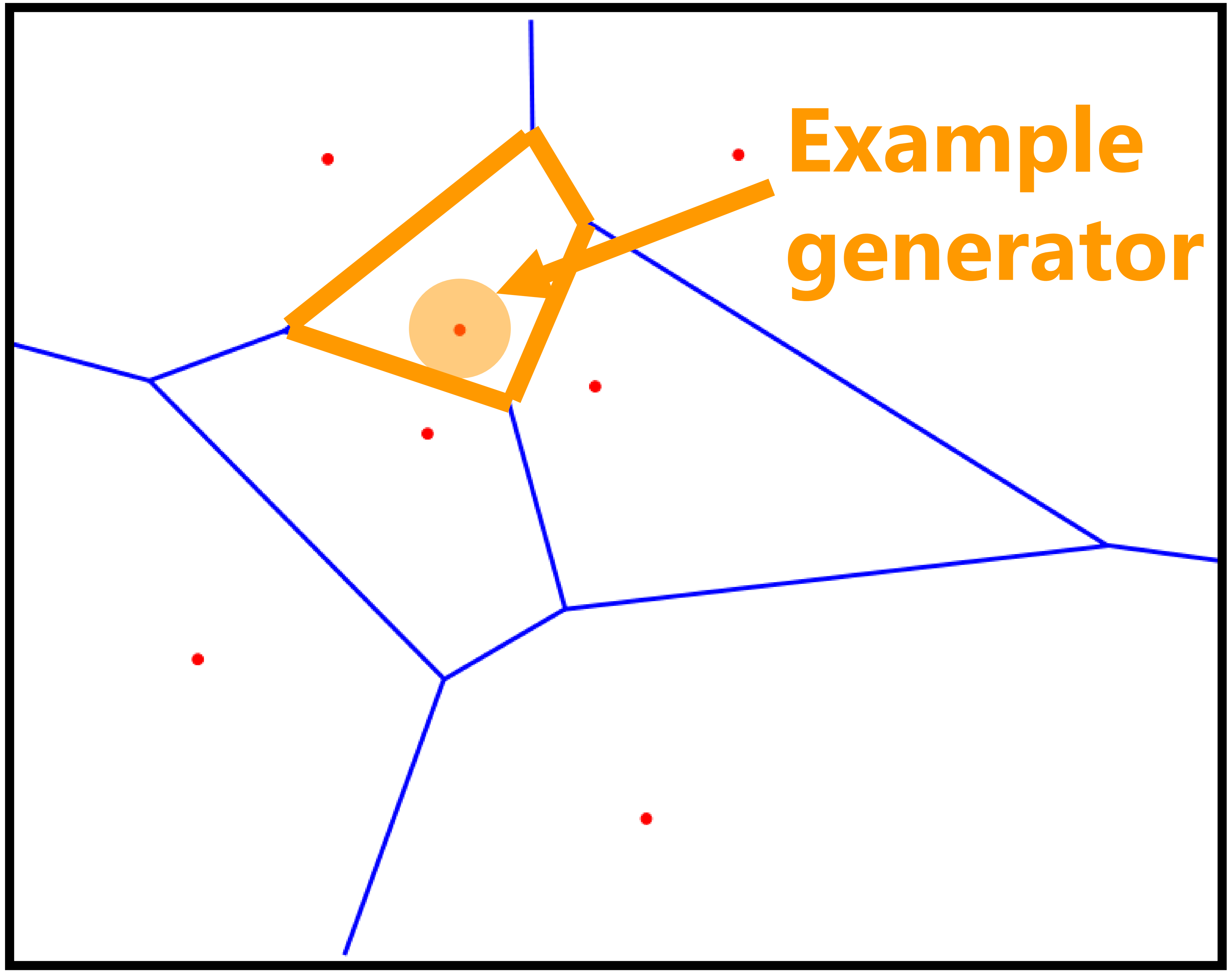
Our method: **Implicitly** construct **Voronoi cells**



Voronoi cell:

A set of points closer to its generator than all the other generators

Our method: **Implicitly** construct **Voronoi cells**



Voronoi cell:

A set of points closer to its generator than all the other generators

Evaluated points represent generators

Exploration

How do you prefer the larger cells
without constructing Voronoi cells?

Exploration

How do you prefer the larger cells
without constructing Voronoi cells?

Use **Voronoi bias**

Exploration: Don't construct Vcells, use **Voronoi bias**

$$x_{t+1} \sim \mathrm{Unif}(\mathcal{X}) \Longrightarrow P(x_{t+1} \in C) = \frac{\mu(C)}{\mu(\mathcal{X})}$$

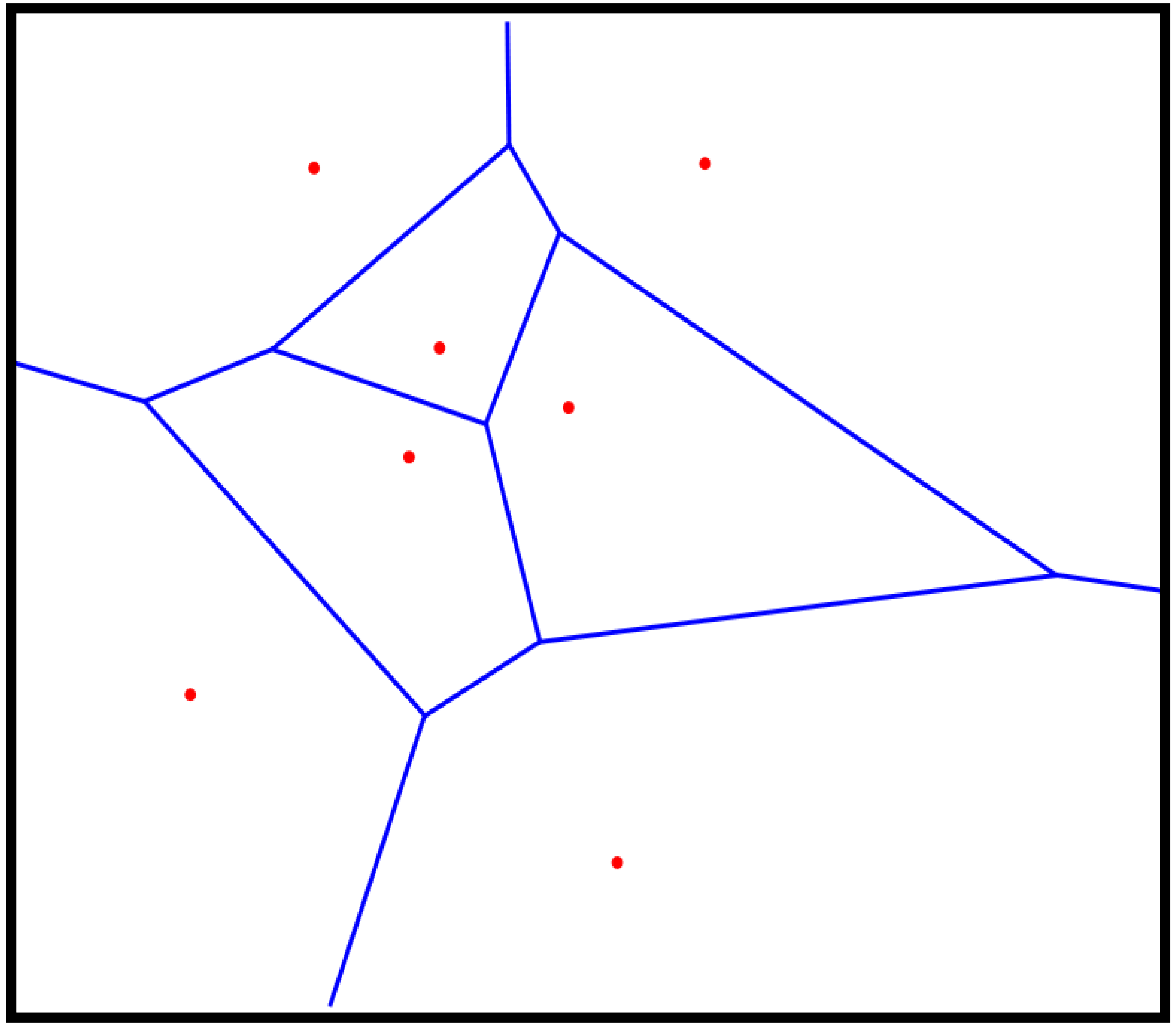
Exploration: Don't construct Vcells, use **Voronoi bias**

Volume of **a cell**

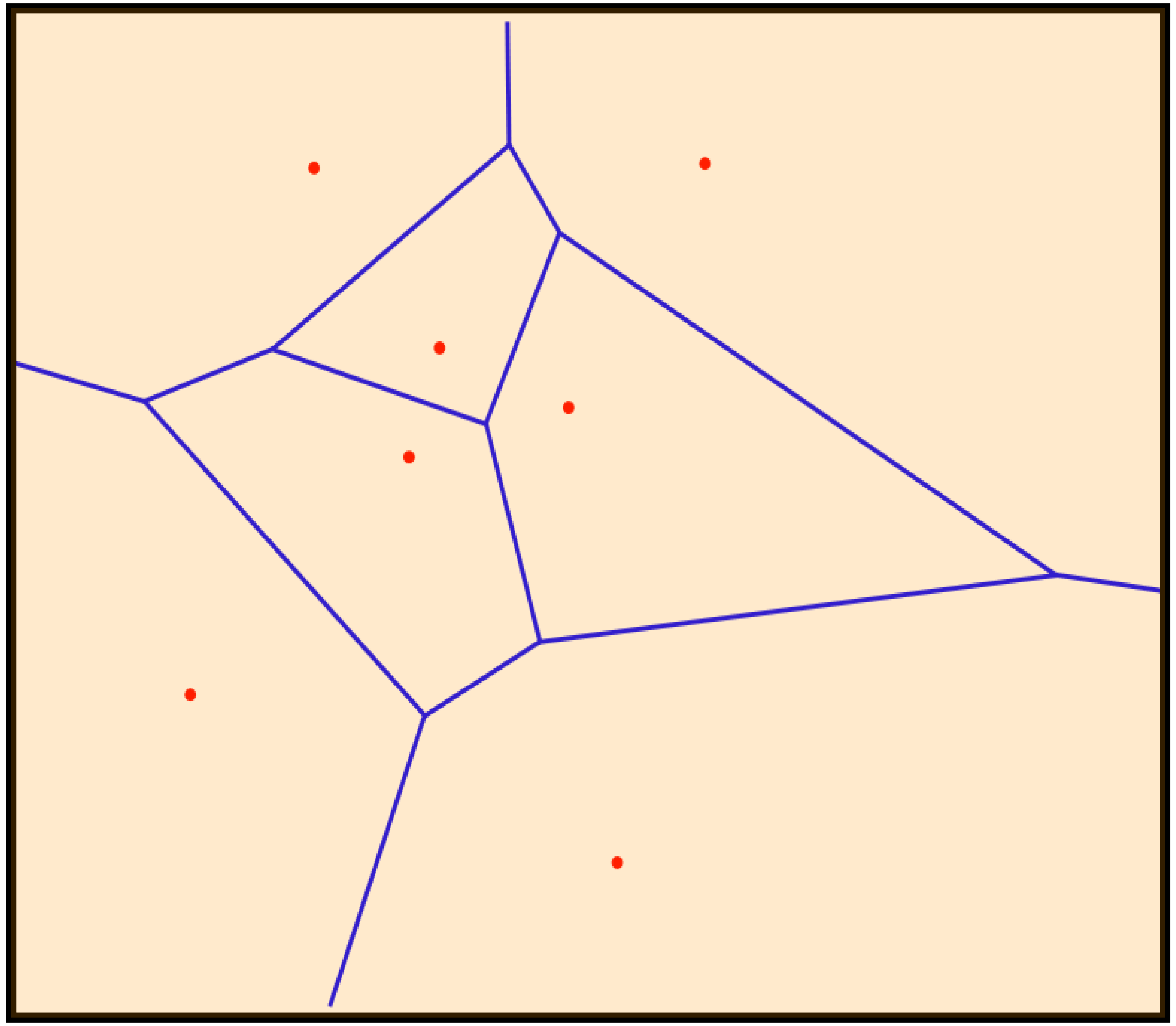
$$x_{t+1} \sim \mathrm{Unif}(\mathcal{X}) \Longrightarrow P(x_{t+1} \in C) = \frac{\mu(C)}{\mu(\mathcal{X})}$$

Volume of **the search space**

Exploration: Don't construct Vcells, use **Voronoi bias**

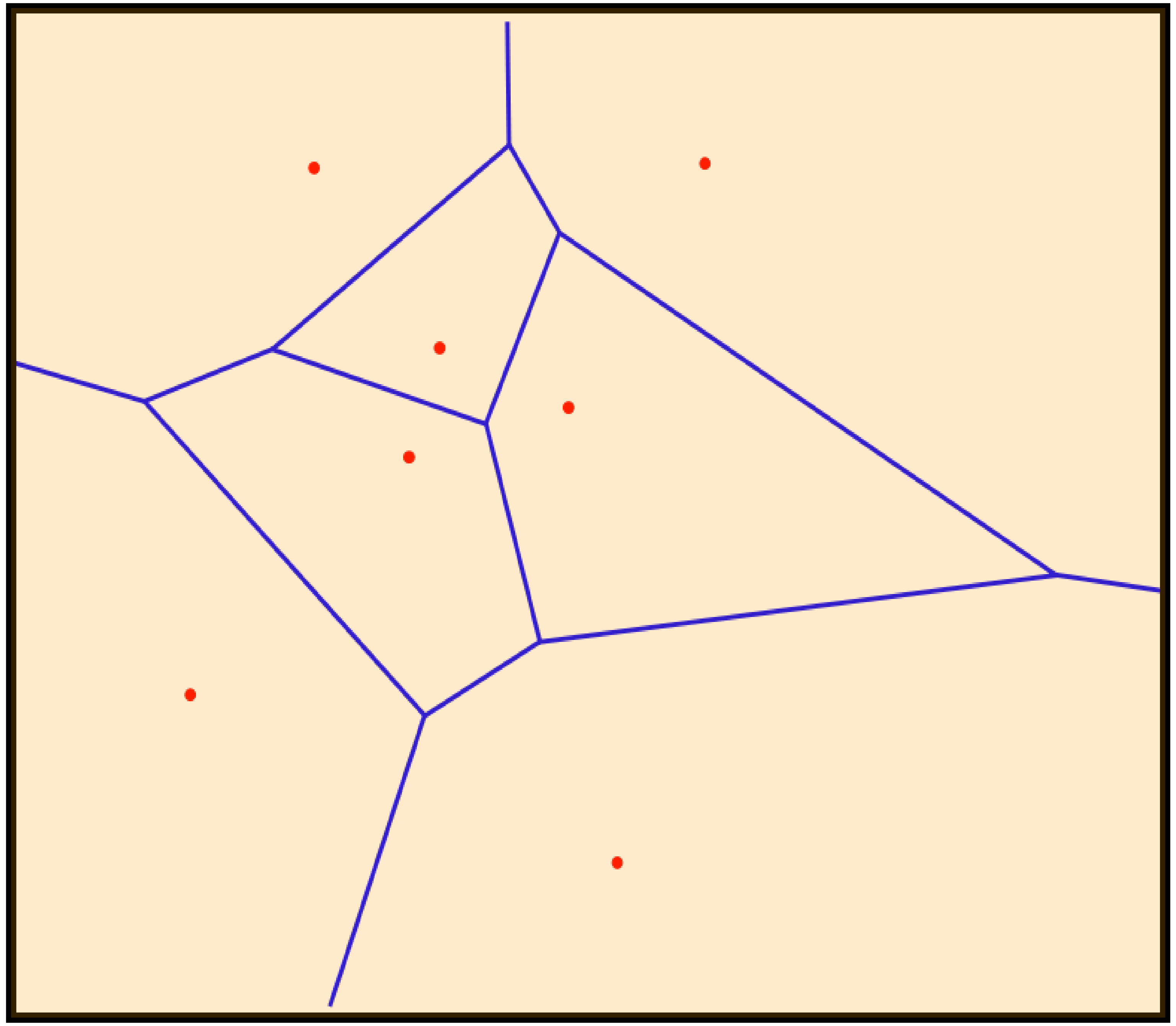


Exploration: Don't construct Vcells, use **Voronoi bias**



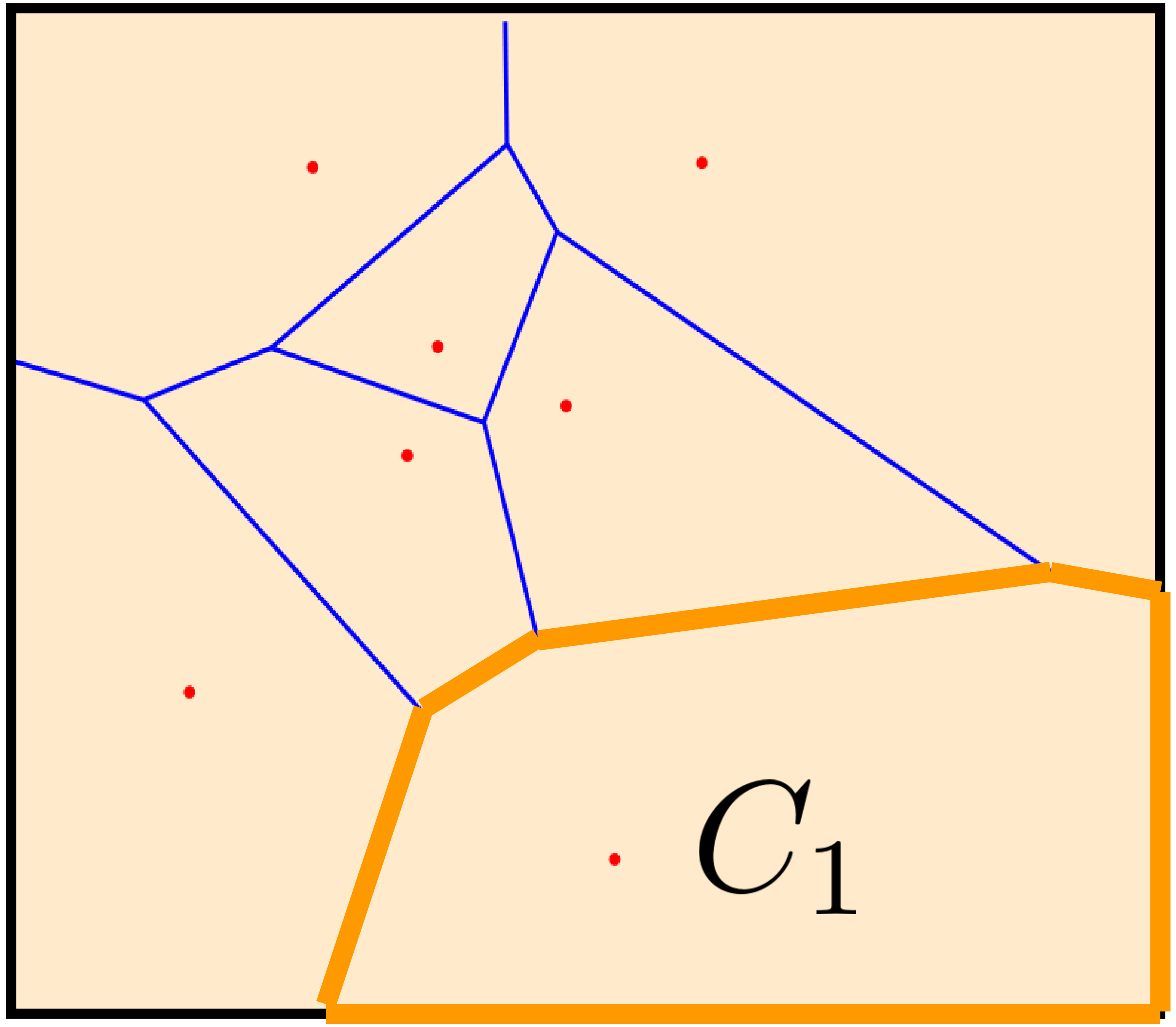
$$\mathrm{Unif}(\mathcal{X})$$

Exploration: Don't construct Vcells, use **Voronoi bias**



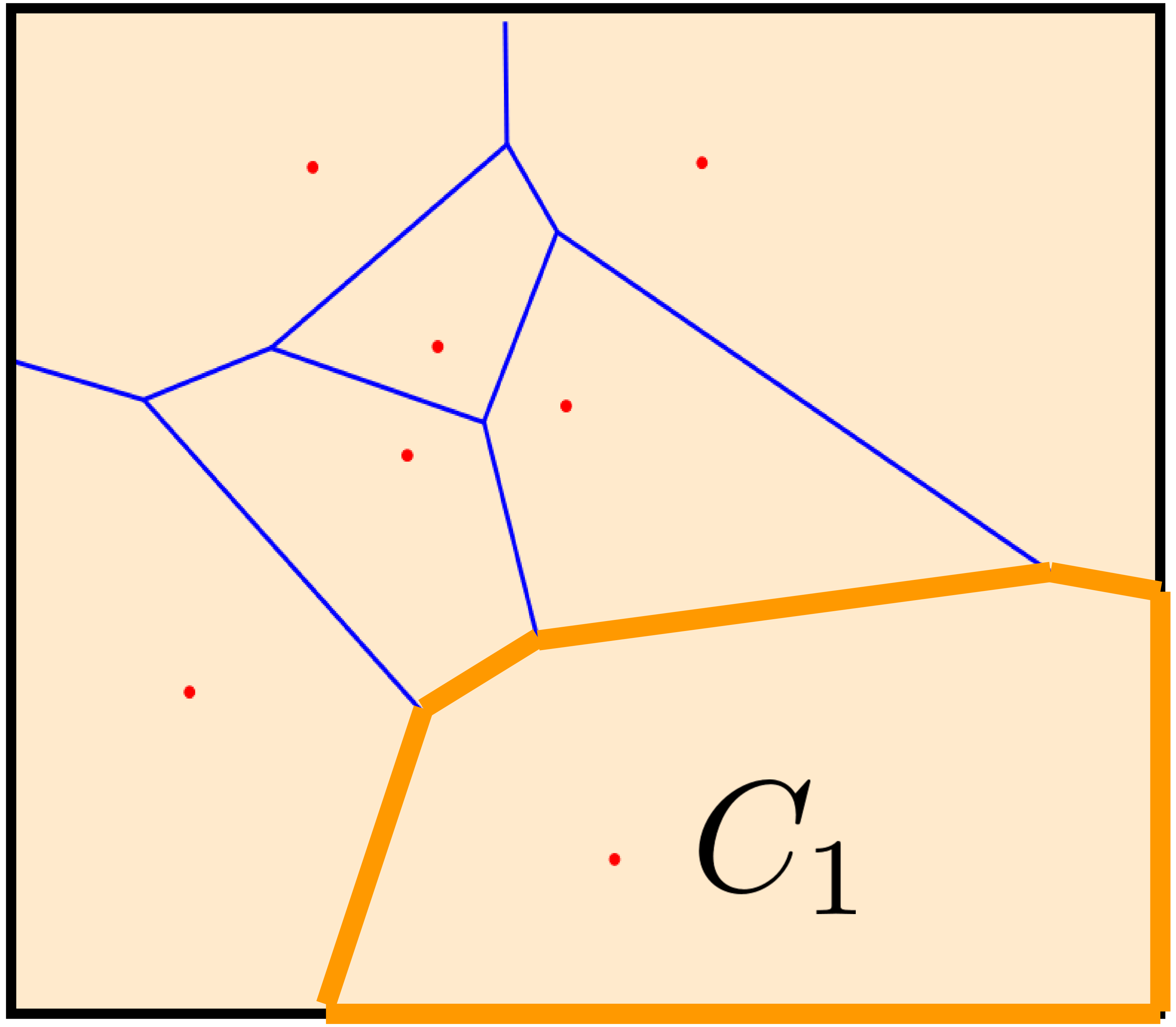
$$x_{t+1} \sim \mathrm{Unif}(\mathcal{X})$$

Exploration: Don't construct Vcells, use **Voronoi bias**



$$x_{t+1} \sim \mathrm{Unif}(\mathcal{X})$$

Exploration: Don't construct Vcells, use **Voronoi bias**



$$x_{t+1} \sim \mathrm{Unif}(\mathcal{X})$$

$$P(x_{t+1} \in C_1) = \frac{\mathrm{Area}(C_1)}{\mathrm{Area}(\mathcal{X})}$$

Exploitation

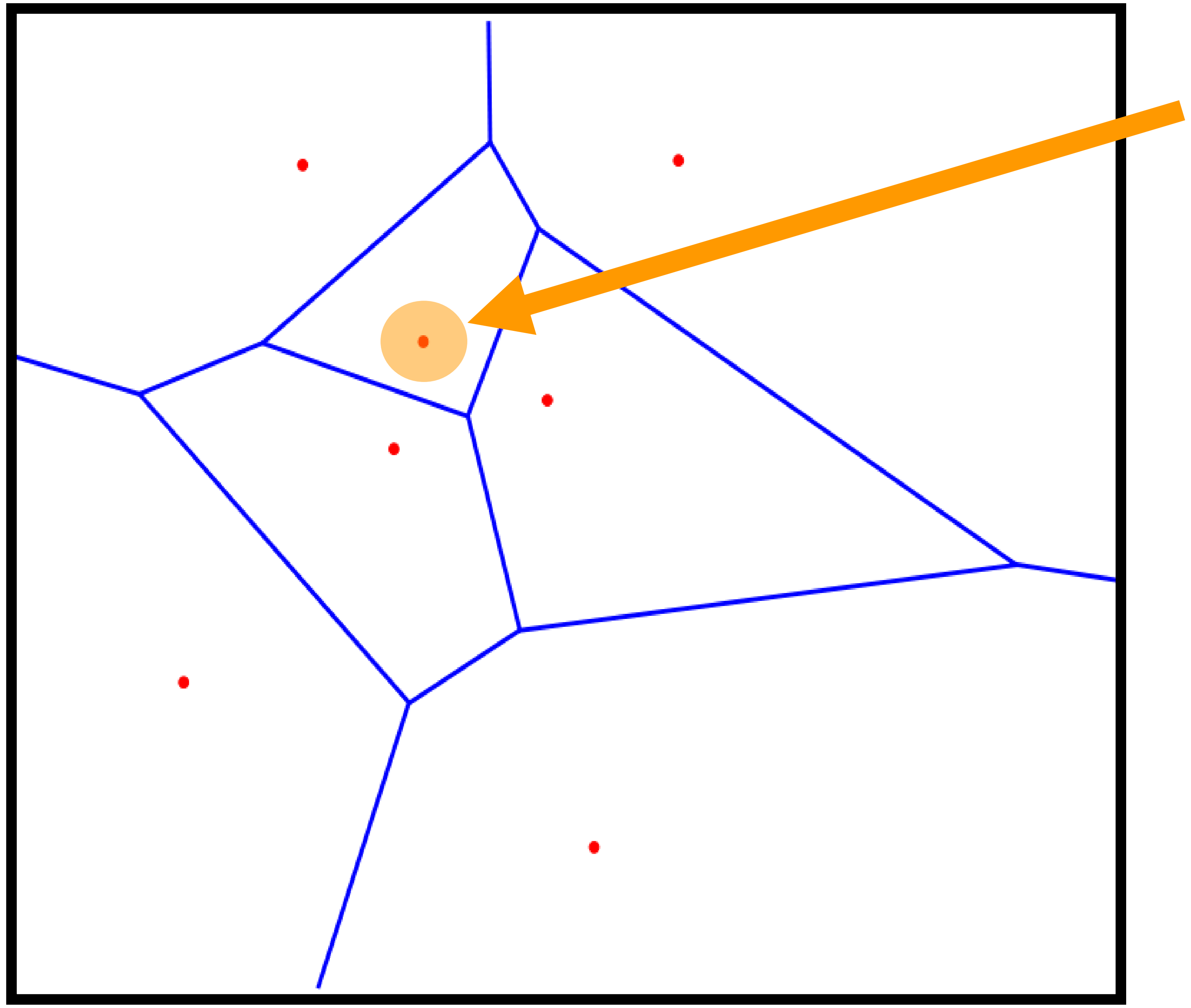
How do you select the cell with the highest value **without constructing Voronoi cells?**

Exploitation

How do you select the cell with the highest value
without constructing Voronoi cells?

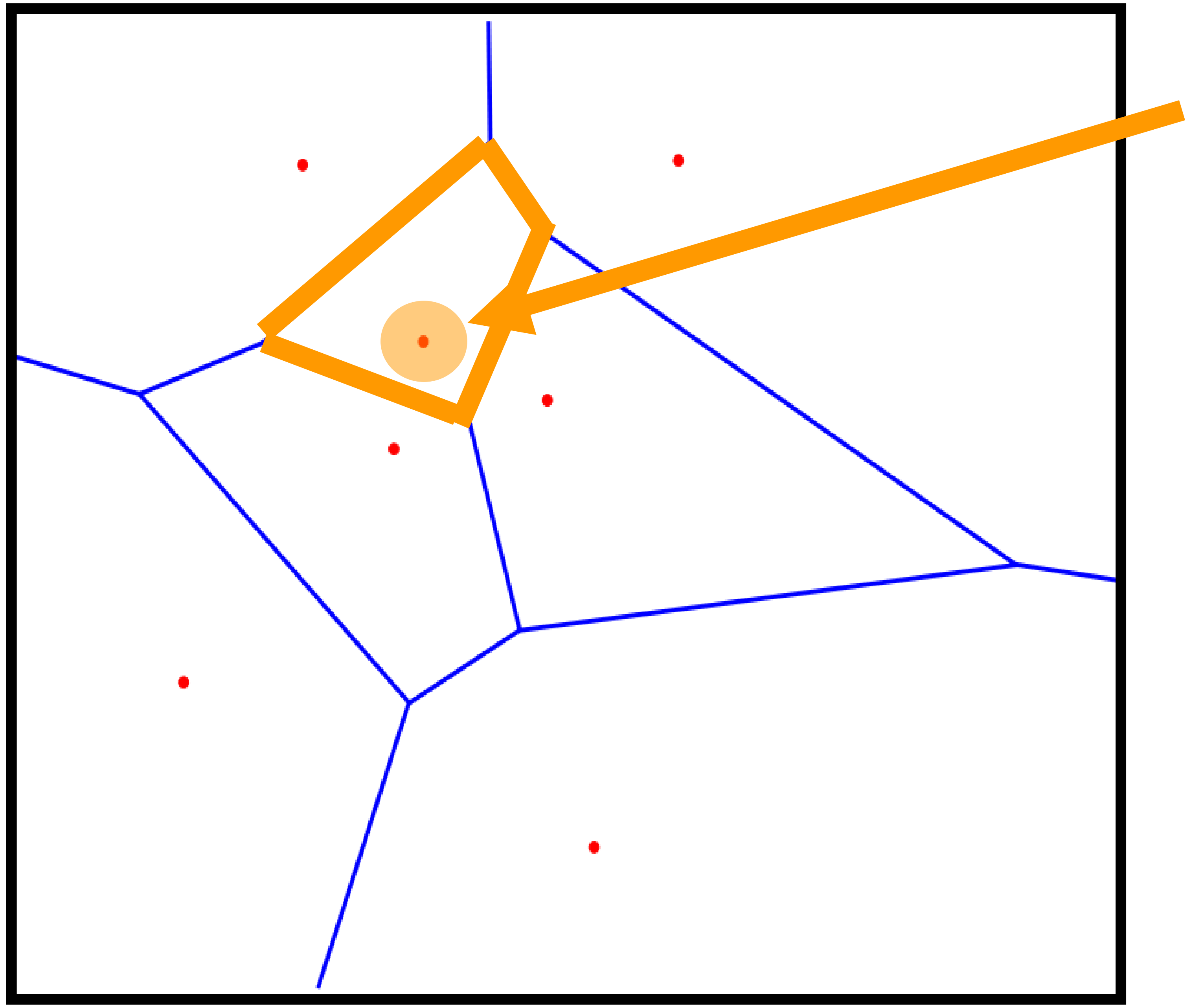
Use **rejection sampling**

Use **rejection sampling** to do exploitation



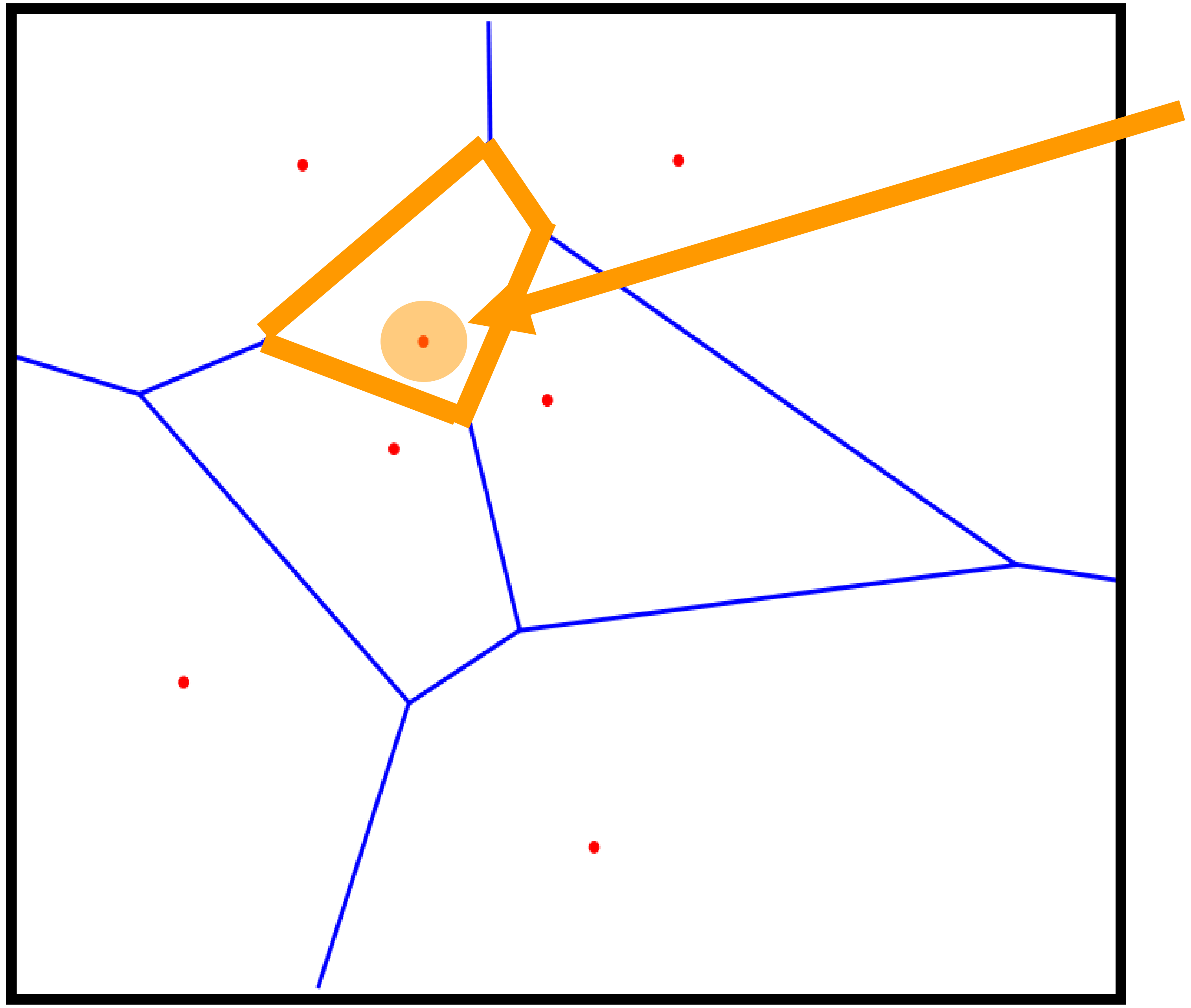
Assume best point found so far

Use **rejection sampling** to do exploitation



Assume best point found so far

Use **rejection sampling** to do exploitation

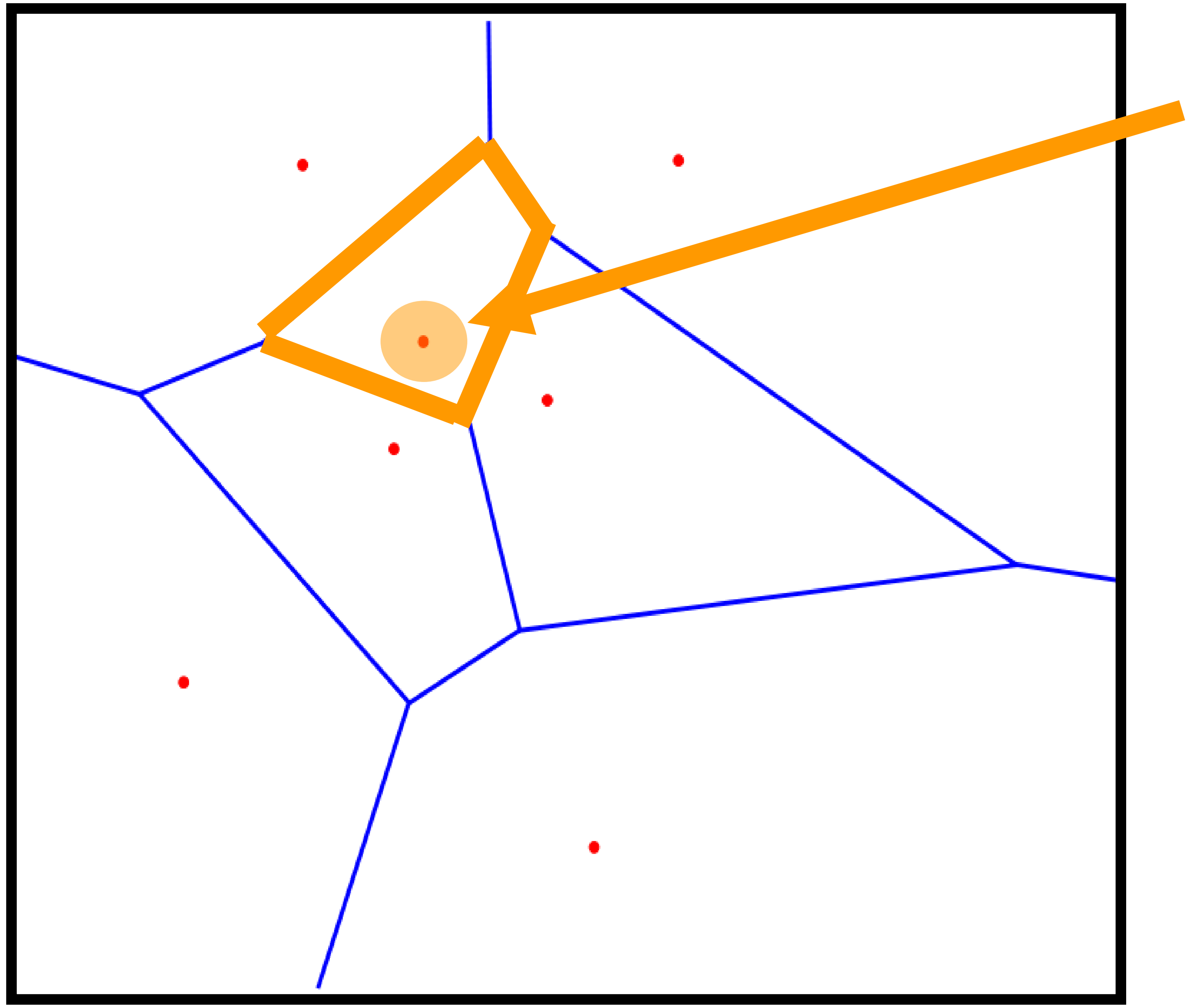


Assume best point found so far

Randomly Sample:

$$x_{t+1} \sim P(\mathcal{X})$$

Use **rejection sampling** to do exploitation



Assume best point found so far

Randomly Sample:

$$x_{t+1} \sim P(\mathcal{X})$$

Reject until the **sampled point is closest to the best point**

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot,\cdot), n)$$

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot,\cdot), n)$$

Search space

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot, \cdot), n)$$

Search space

Exploration parameter

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot,\cdot), n)$$

Search space

Exploration parameter

Distance metric

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot,\cdot), n)$$

Search space

Exploration parameter

Distance metric

Budget

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot,\cdot), n)$$

With probability ω

$$x_{t+1} \sim \mathrm{UniformRandomlySample}(\mathcal{X})$$

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot,\cdot), n)$$

With probability ω **Exploration**

$$x_{t+1} \sim \mathrm{UniformRandomlySample}(\mathcal{X})$$

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot,\cdot), n)$$

With probability ω **Exploration**

$$x_{t+1} \sim \mathrm{UniformRandomlySample}(\mathcal{X})$$

With probability $1-\omega$

$$x_{t+1} \sim \mathrm{SampleBestVCell}(d(\cdot,\cdot), x_{1:t})$$

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot,\cdot), n)$$

With probability ω

Exploration

$$x_{t+1} \sim \mathrm{UniformRandomlySample}(\mathcal{X})$$

With probability $1-\omega$

$$x_{t+1} \sim \mathrm{SampleBestVCell}(d(\cdot,\cdot), x_{1:t})$$

Exploitation

VOO: Voronoi optimistic optimization

$$\mathrm{VOO}(\mathcal{X}, \omega, d(\cdot,\cdot), n)$$

With probability ω

Exploration

$$x_{t+1} \sim \mathrm{UniformRandomlySample}(\mathcal{X})$$

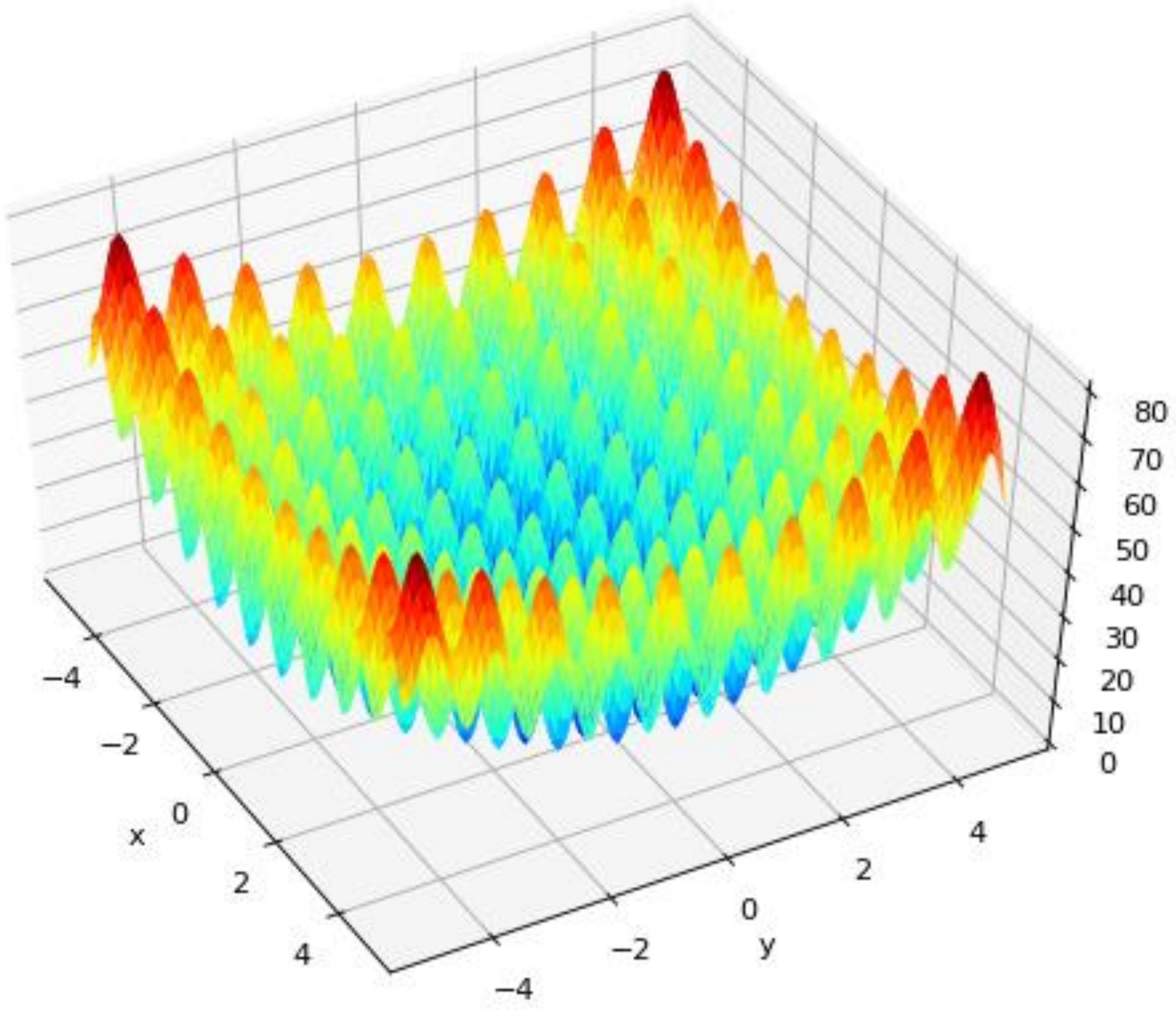
With probability $1 - \omega$

$$x_{t+1} \sim \mathrm{SampleBestVCell}(d(\cdot,\cdot), x_{1:t})$$

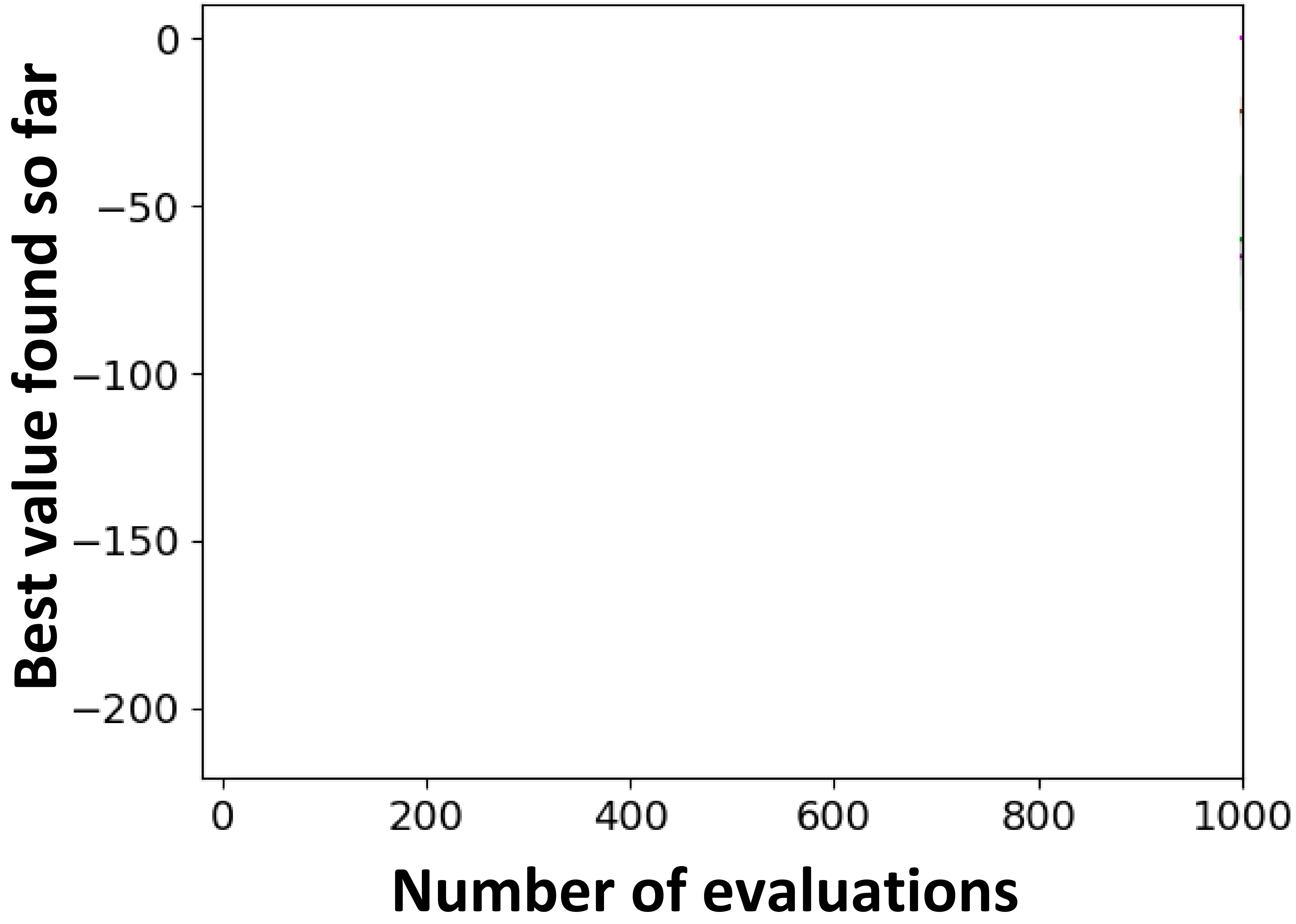
Evaluate $f(x_{t+1})$

Exploitation

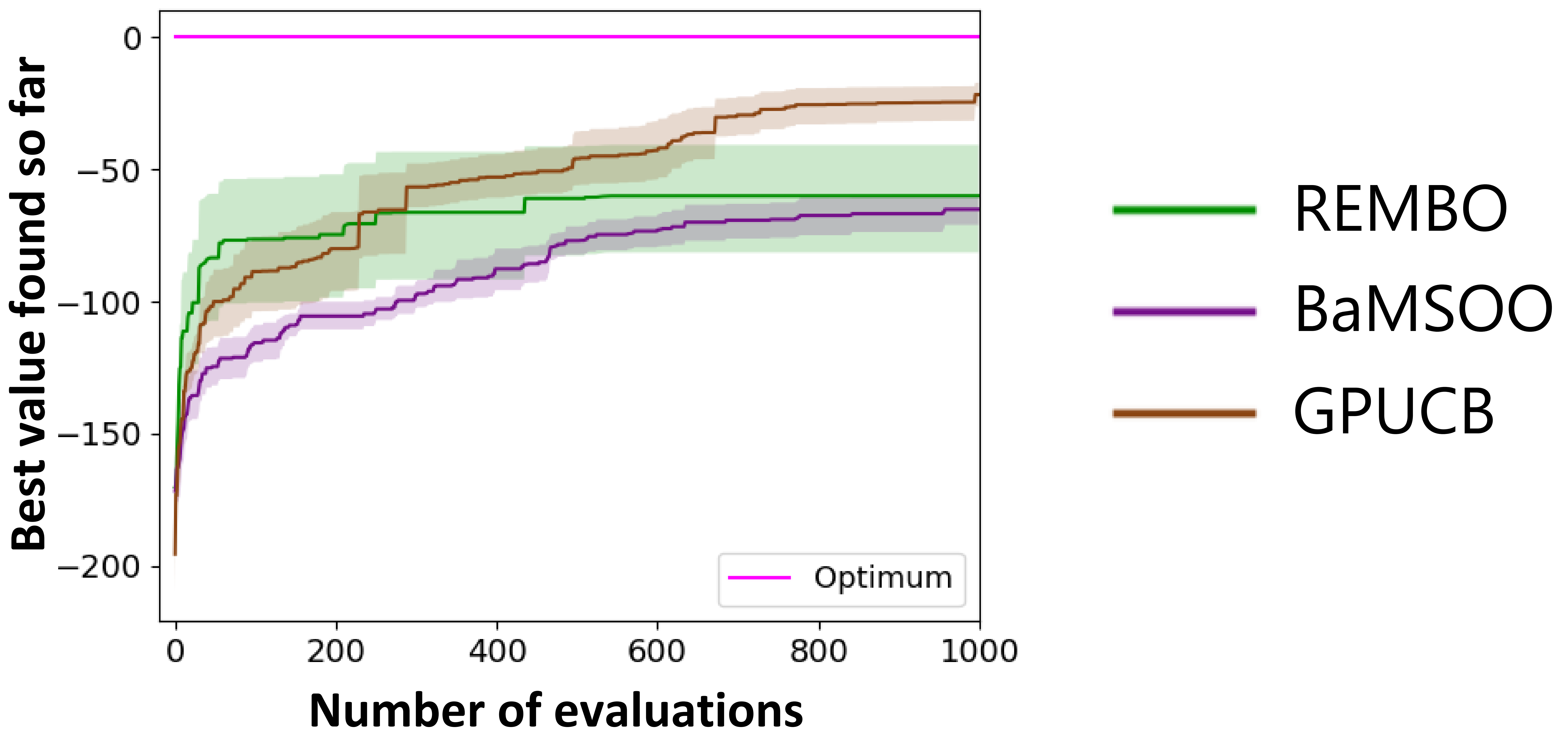
Optimizing Rastrigin function



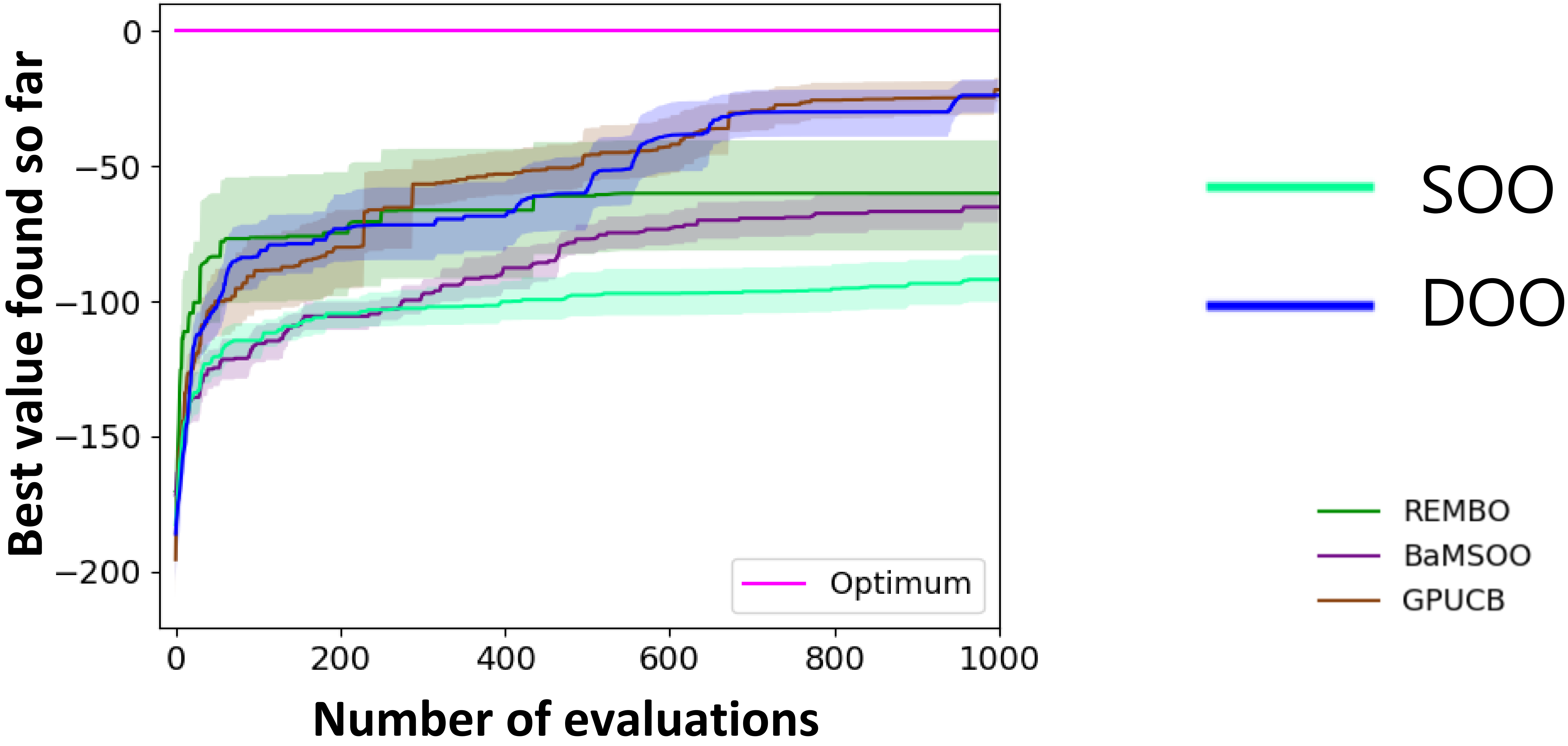
Optimizing **10 dimensional** Rastrigin function



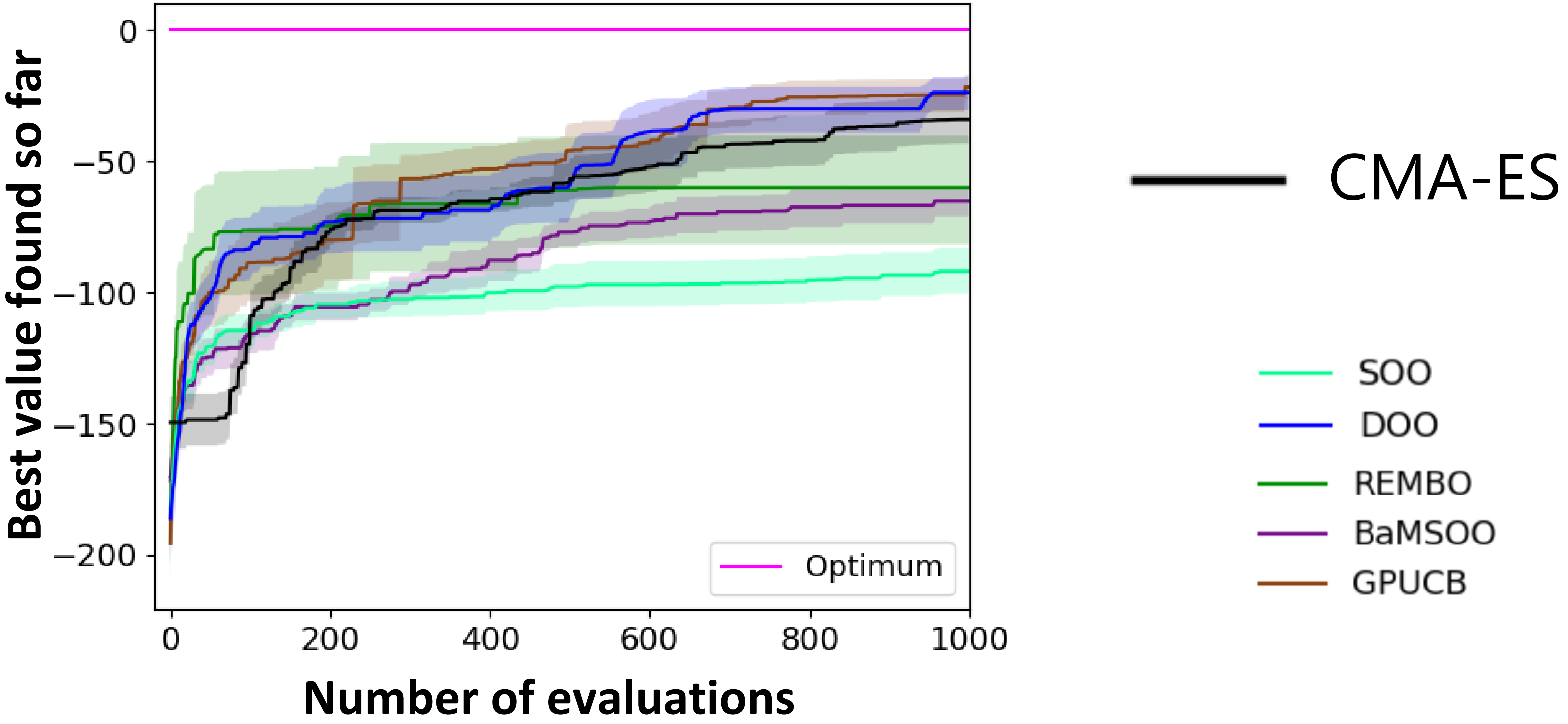
10D – Bayesian optimization methods



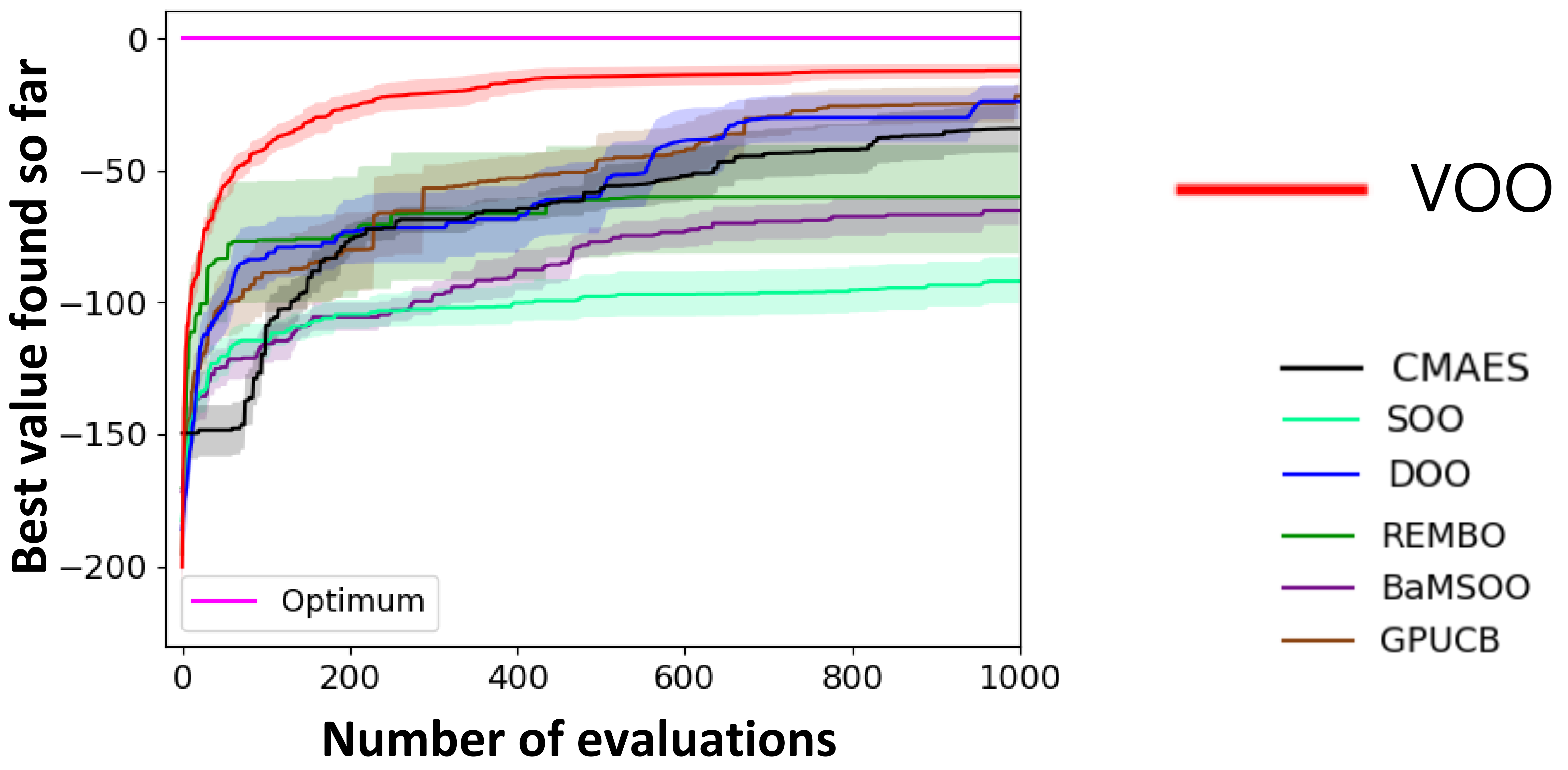
10D – Hierarchical partitioning methods



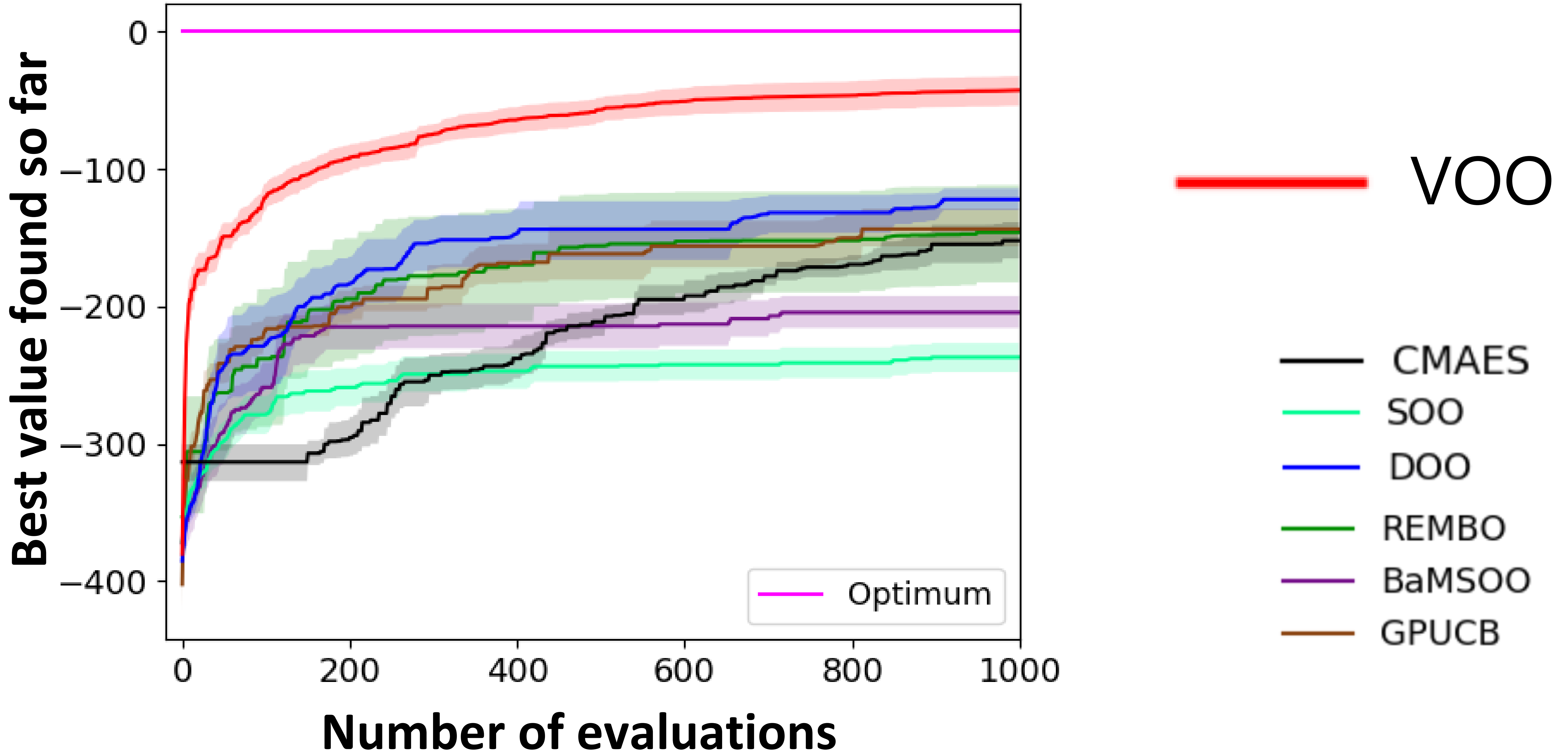
10D – Evolutionary algorithm



10D – Voronoi optimistic optimization



20D – even more drastic difference



From BBFO to MCTS

- So that was when **planning horizon is 1**
- What about **planning horizon >= 1**, for **deterministic environments?**

VOO applied to Trees (VOOT): high-level idea

Assign a VOO agent at each node

VOO applied to Trees (VOOT): high-level idea

Assign a VOO agent at each node

Solve the action optimization problem using the **estimated Q-function**

$$\max_{a \in \mathcal{A}} \hat{Q}(s, a)$$

Our main theoretical result

How many simulations do we need to guarantee

$$V_\star(s_0) - \hat{V}_N(s_0) \leq \eta, \eta > 0$$

Our main theoretical result

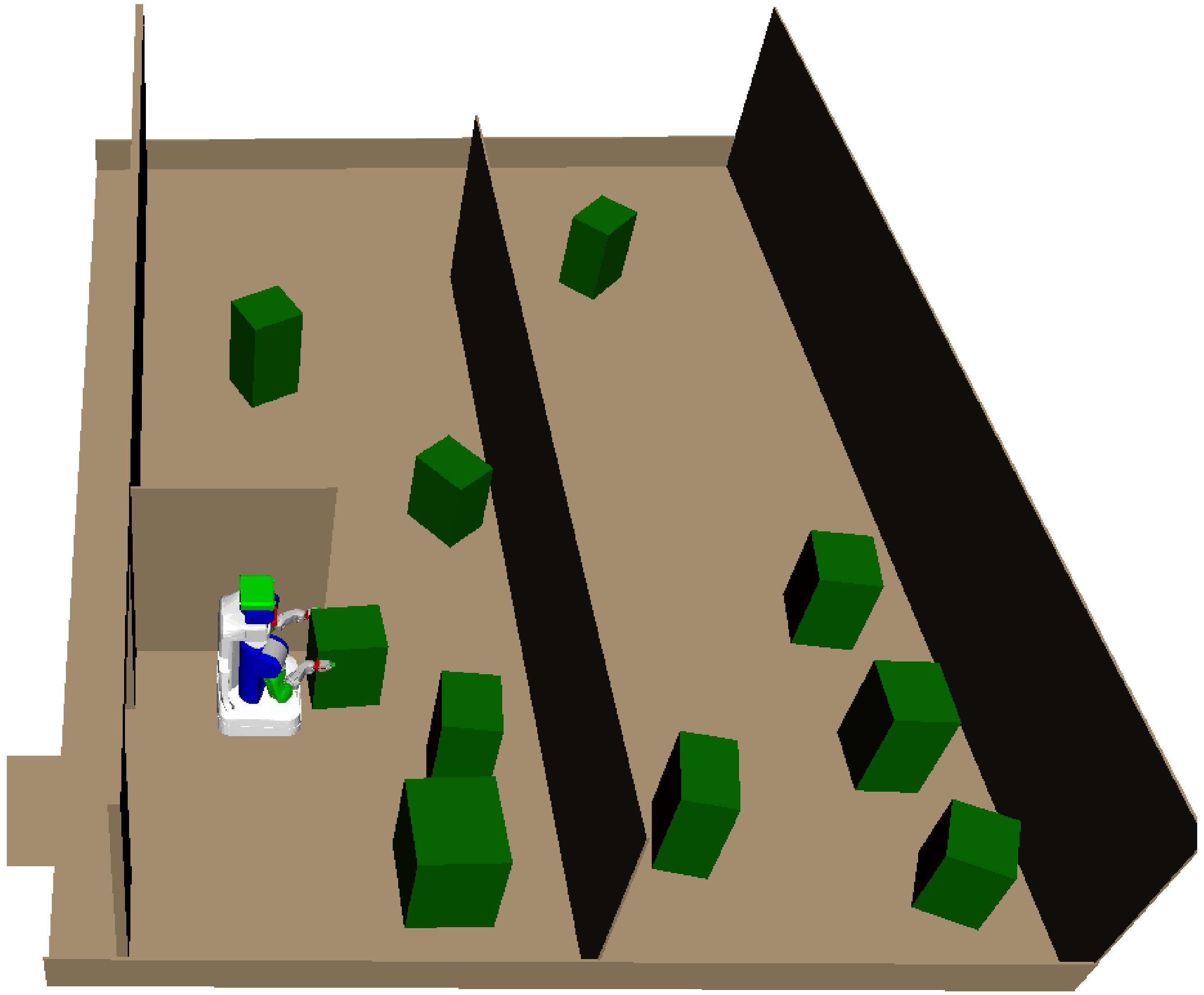
How many simulations do we need to guarantee

$$V_\star(s_0) - \hat{V}_N(s_0) \leq \eta, \eta > 0$$

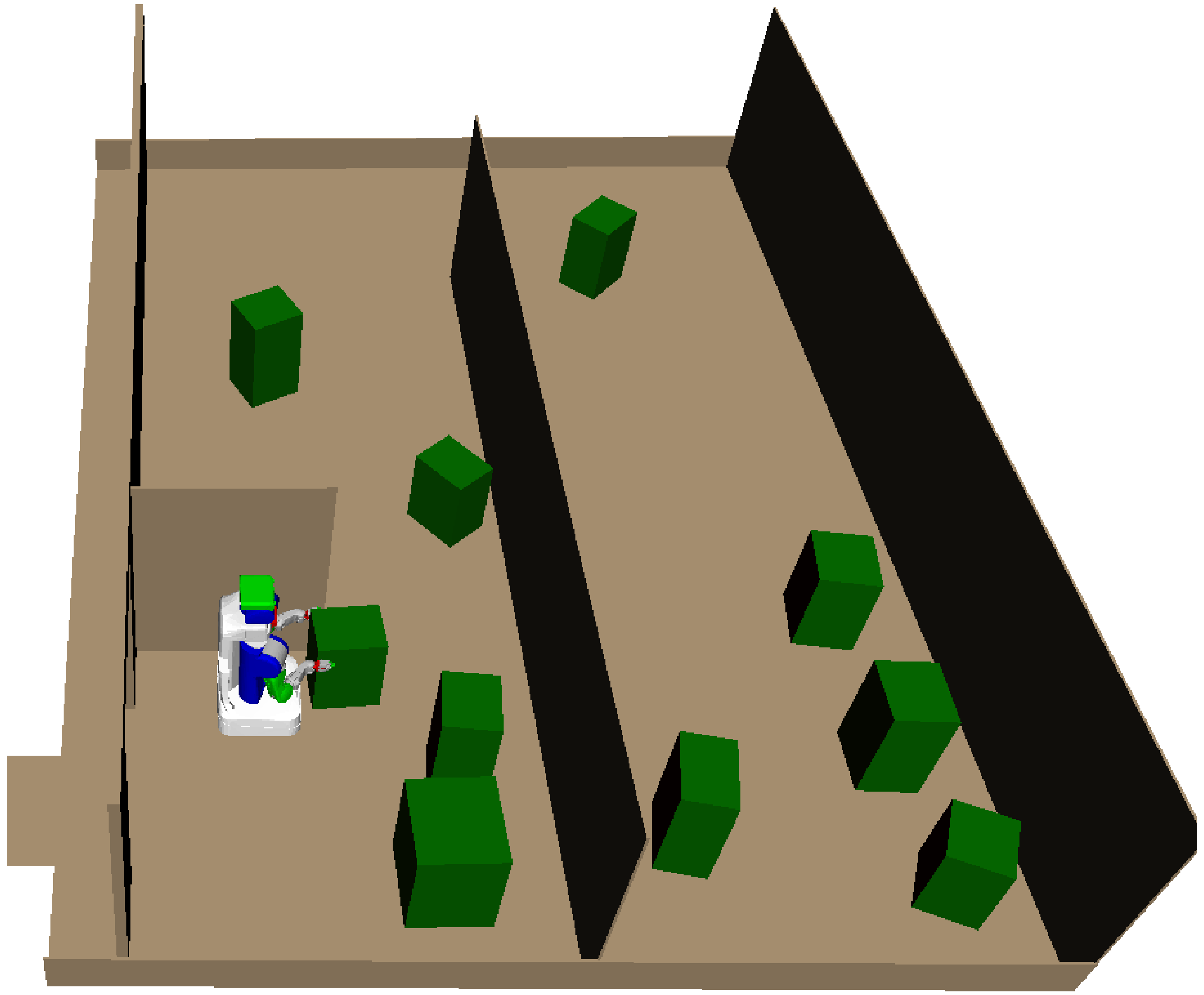
Under mild assumptions, $\mathbb{E}[N] = m^h$

$$m = \log\left(\frac{\eta(1-\gamma)}{2L\delta_{max}C_{max}}\right) \cdot \min(G_{\lambda,\omega}, K_{\nu,\omega,\lambda})$$

Packing problem

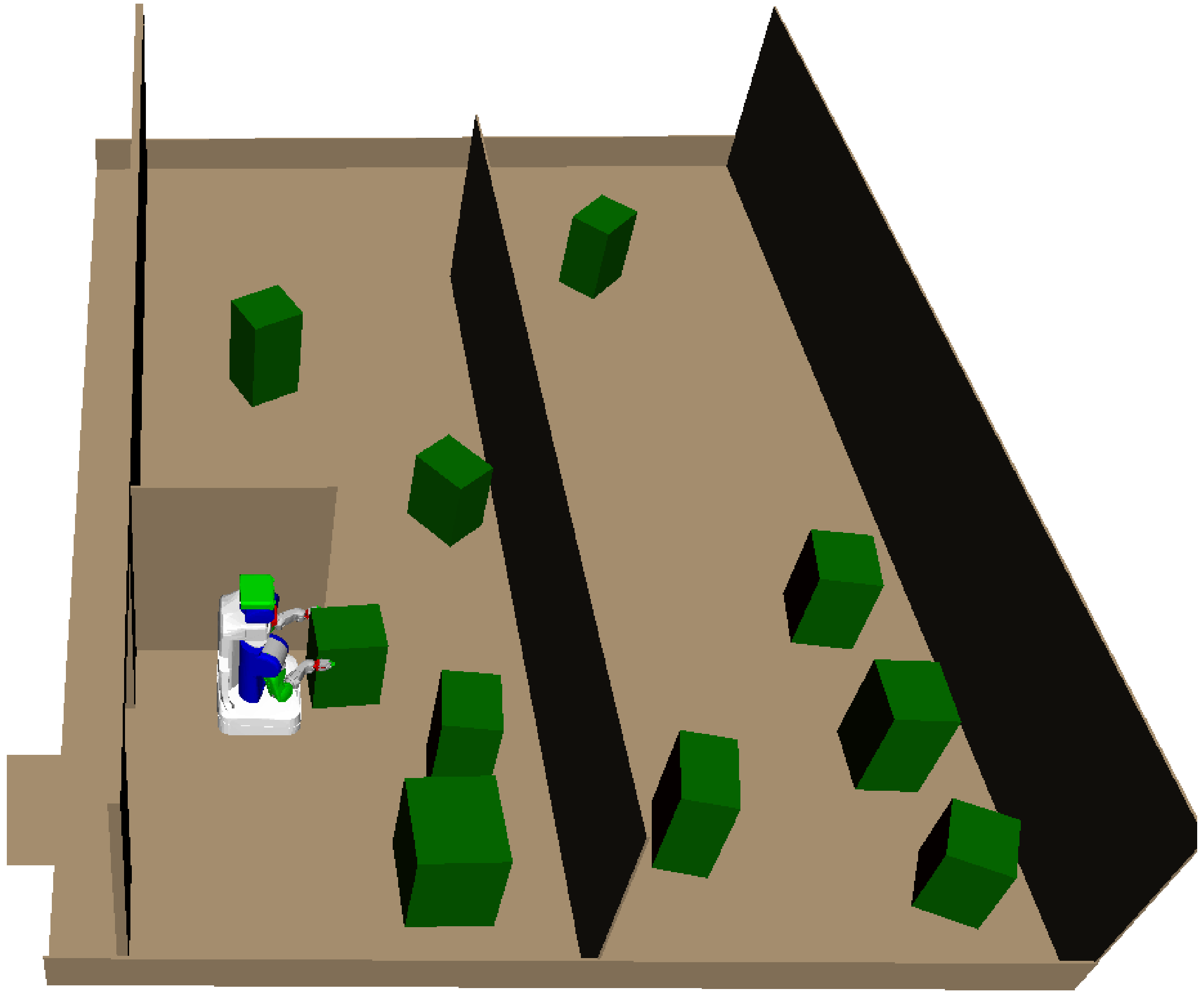


Packing problem



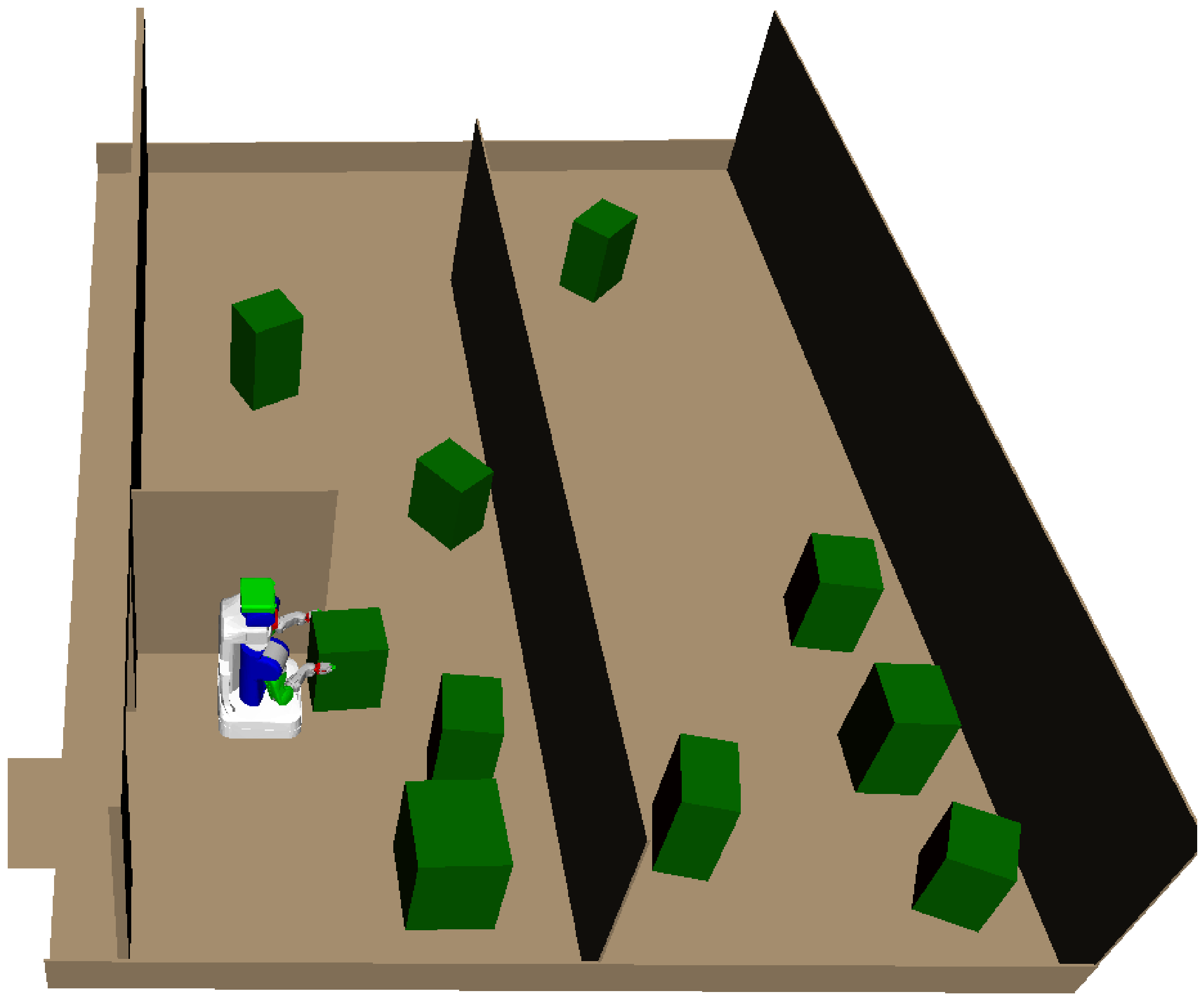
- **Actions**: placements for three boxes (9 dim)

Packing problem



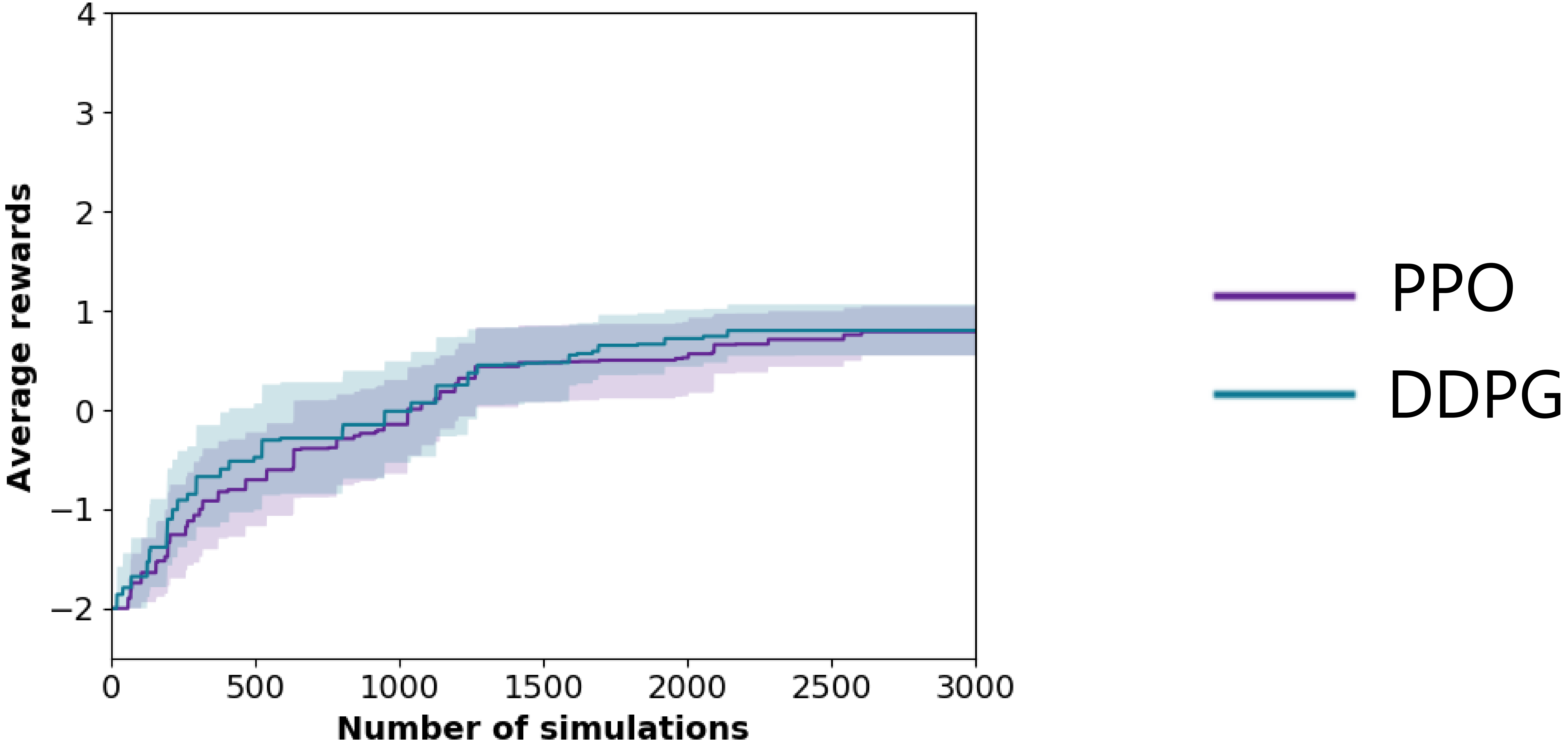
- **Actions**: placements for three boxes (9 dim)
- **Reward function**:
  - 1/d if feasible,
  - r_min otherwise

Packing problem

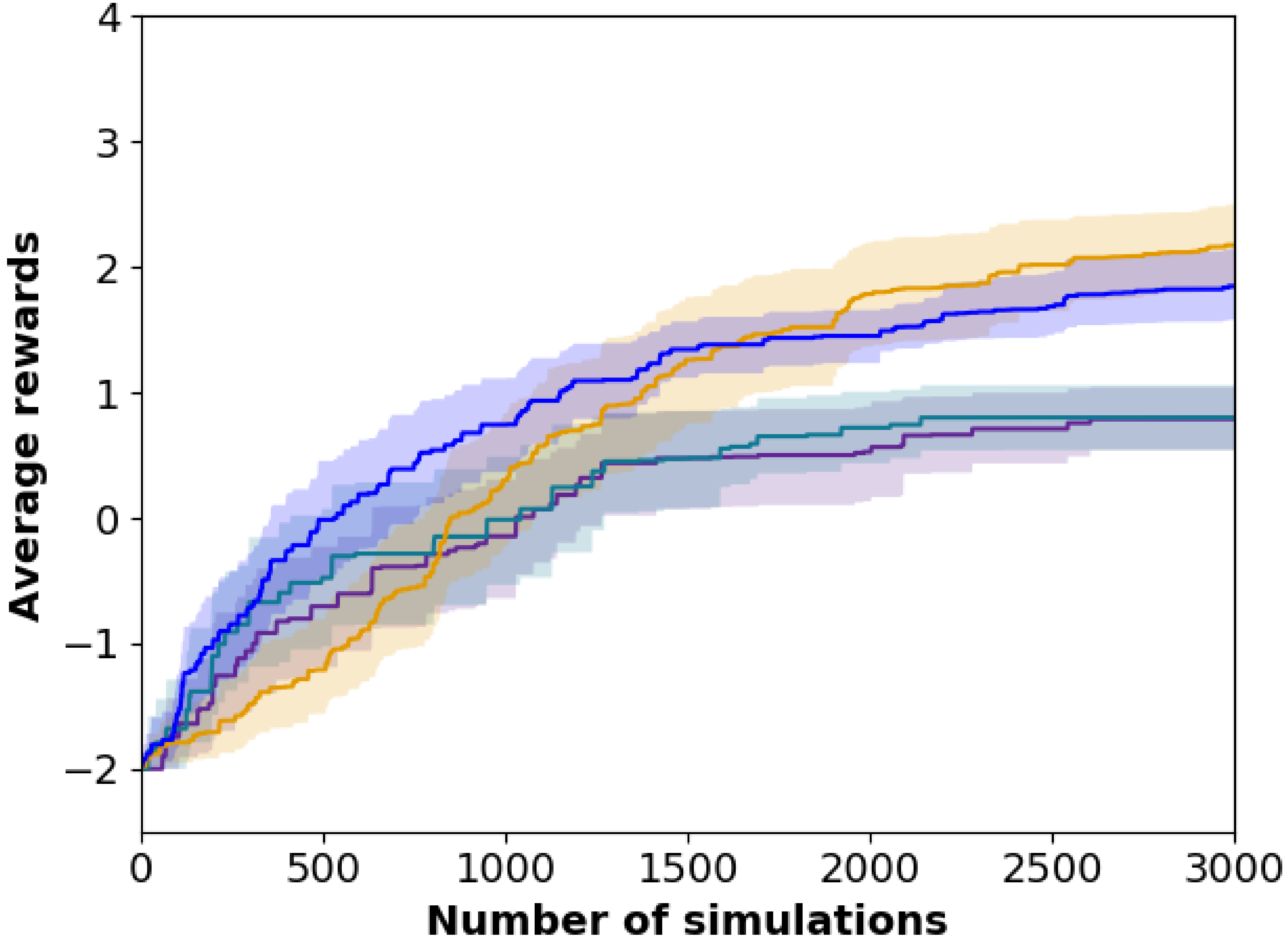


- **Actions**: placements for three boxes (9 dim)
- **Reward function**:
  - 1/d if feasible,
  - r_min otherwise
- **Total boxes to pack**: 20

State-of-the-art **RL algorithms**

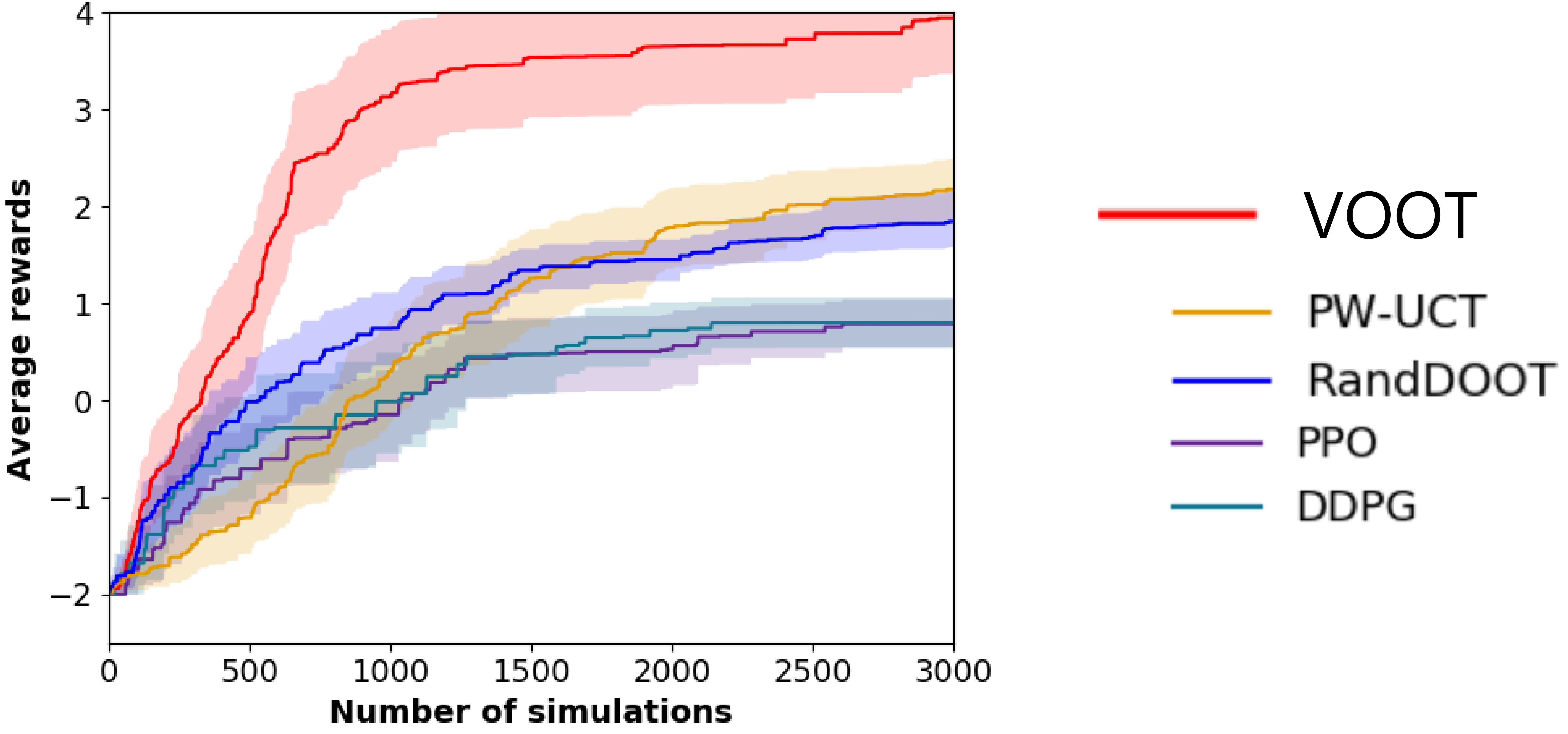


State-of-the-art **continuous-action MCTS**



PW-UCT
DOOT
PPO
DDPG

VOO applied to Trees



Takeaways

Voronoi Optimistic Optimization applied to Trees (VOOT) for **deterministic environment**

- Empirical contribution: improvement over the state-of-the-art, especially in **high-dimensional action-spaces**
- Theoretical contribution: **regret-bound for VOOT**