

# **Expensive black-box combinatorial optimization**

**Probabilistic optimization with few samples**

**Ekhiñe Irurozki**

In collaboration with **Manuel Lopez-Ibanez** (University of Manchester, University of Malaga)

Accepted at **The Genetic and Evolutionary Computation Conference, GECCO21**

Huawei, 7/6/21

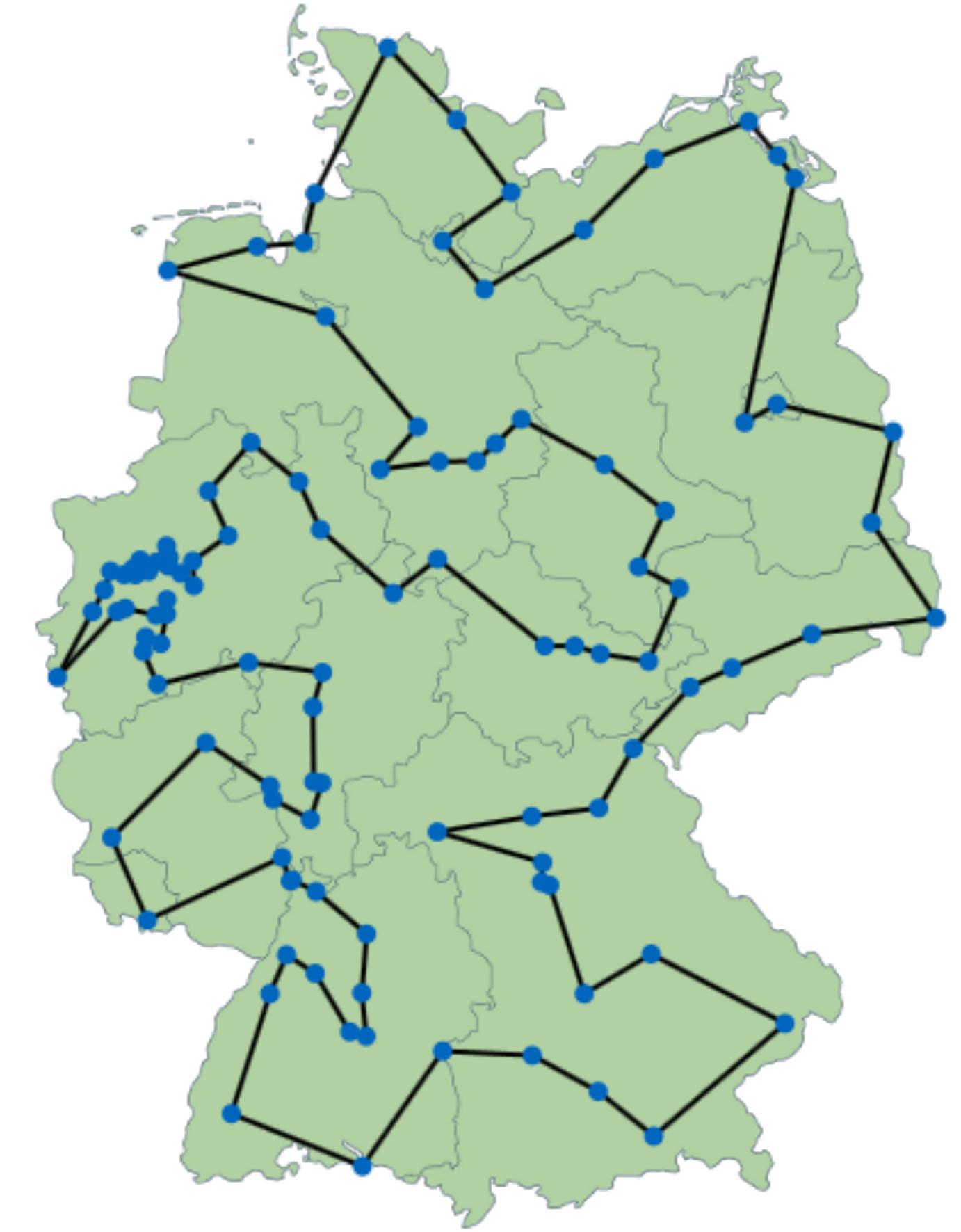
# **Permutation problems**

**A solution is a permutation of objects**

# Permutation problems

A solution is a permutation of objects

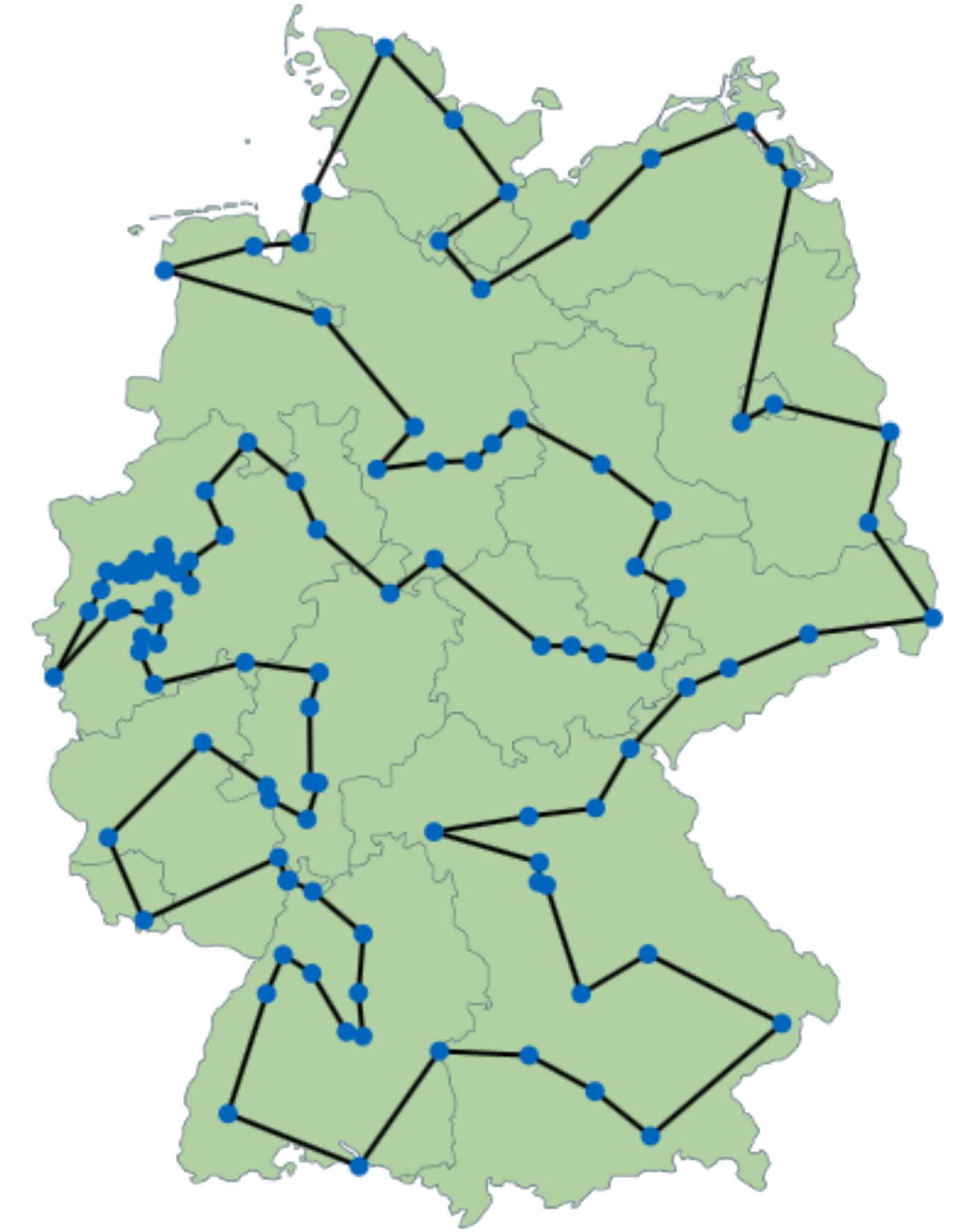
- Traveling Salesman Problem (TSP)



# Permutation problems

A solution is a permutation of objects

- Traveling Salesman Problem (TSP)
- For  $n$  cities there are  $n!$  possible routes
  - $53! \approx 4.27e69$

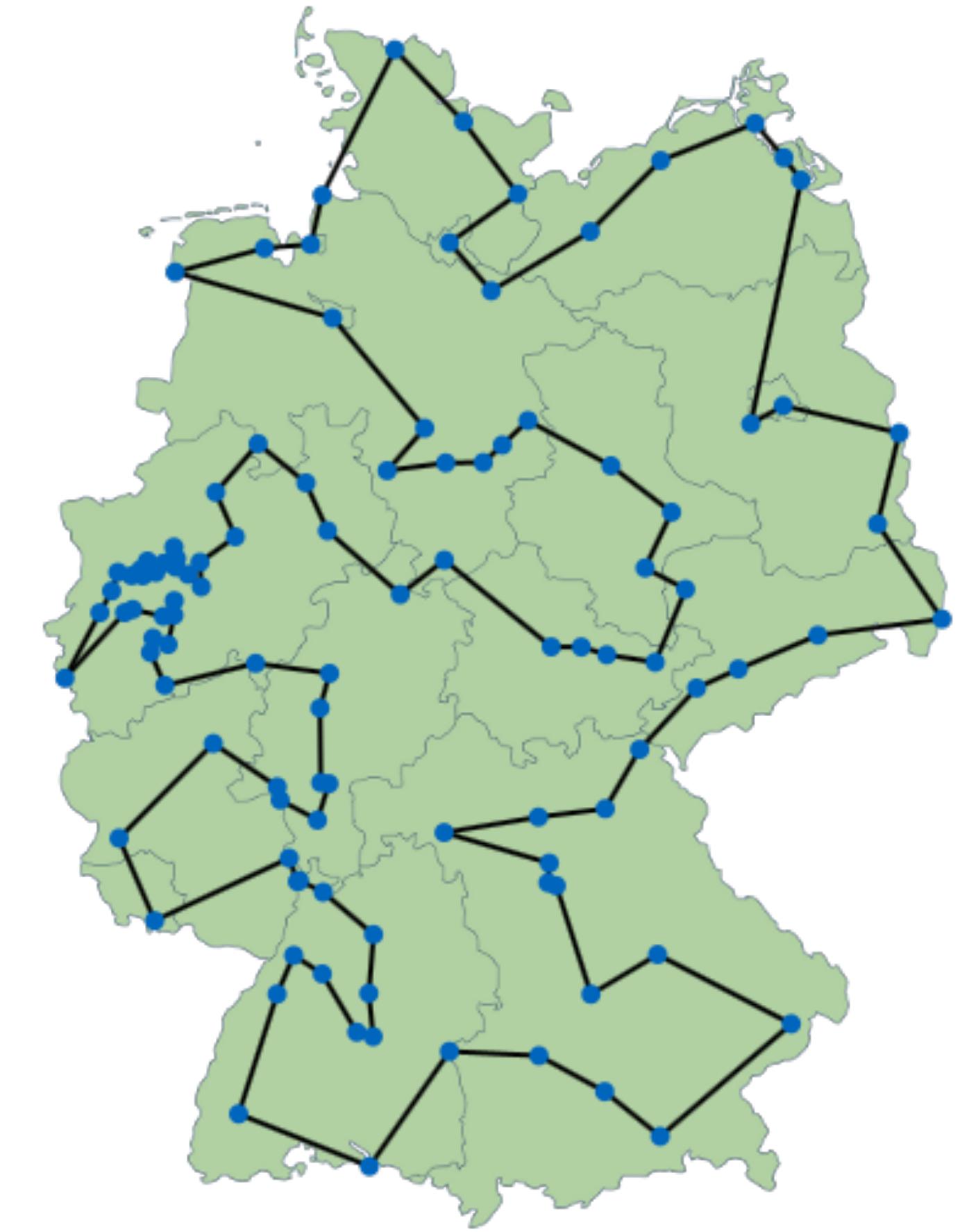


# Permutation problems

A solution is a permutation of objects

- Traveling Salesman Problem (TSP)
- For  $n$  cities there are  $n!$  possible routes
  - $53! \approx 4.27e69 > 1.2e68$

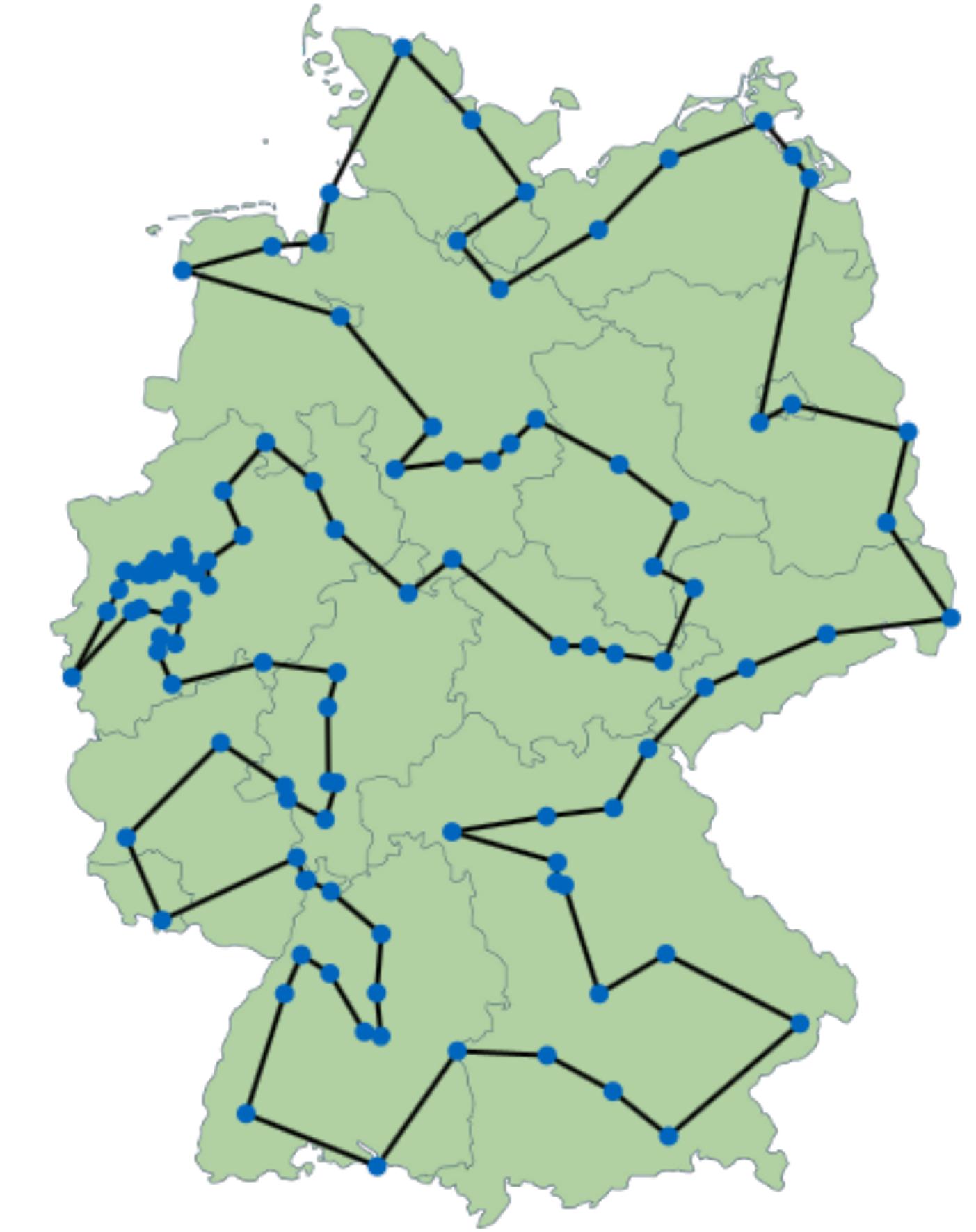
Number of atoms in the galaxy



# Permutation problems

A solution is a permutation of objects

- Traveling Salesman Problem (TSP)
- For  $n$  cities there are  $n!$  possible routes
  - $53! \approx 4.27e69 > 1.2e68$   
Number of atoms in the galaxy
- For 15k towns it takes 22 years of computation to get the optimal tour



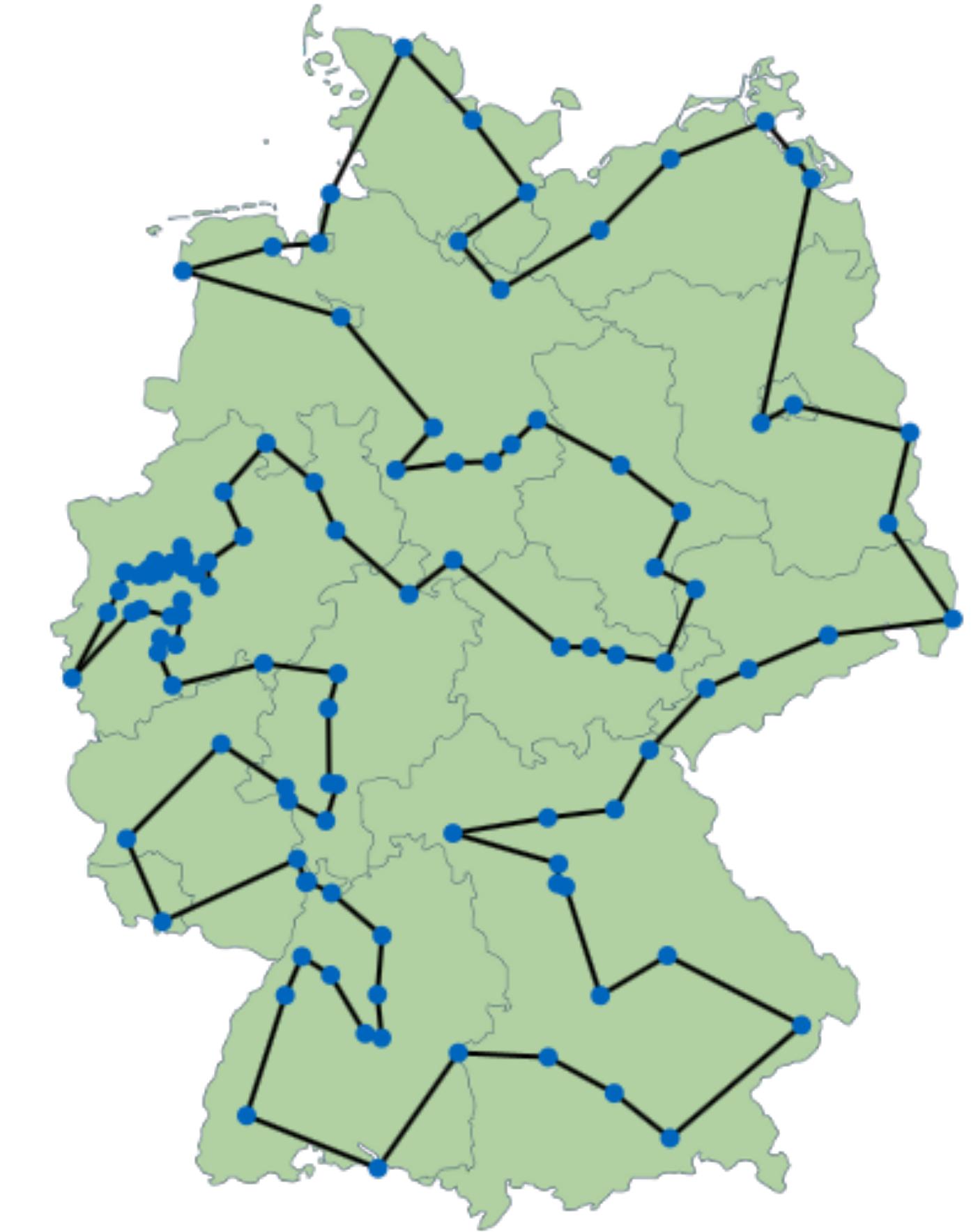
# Permutation problems

A solution is a permutation of objects

- Traveling Salesman Problem (TSP)
- For  $n$  cities there are  $n!$  possible routes
  - $53! \approx 4.27e69 > 1.2e68$

Number of atoms in the galaxy

- For 15k towns it takes 22 years of computation to get the optimal tour
- Combinatorial optimization finds *good* solutions *quickly*



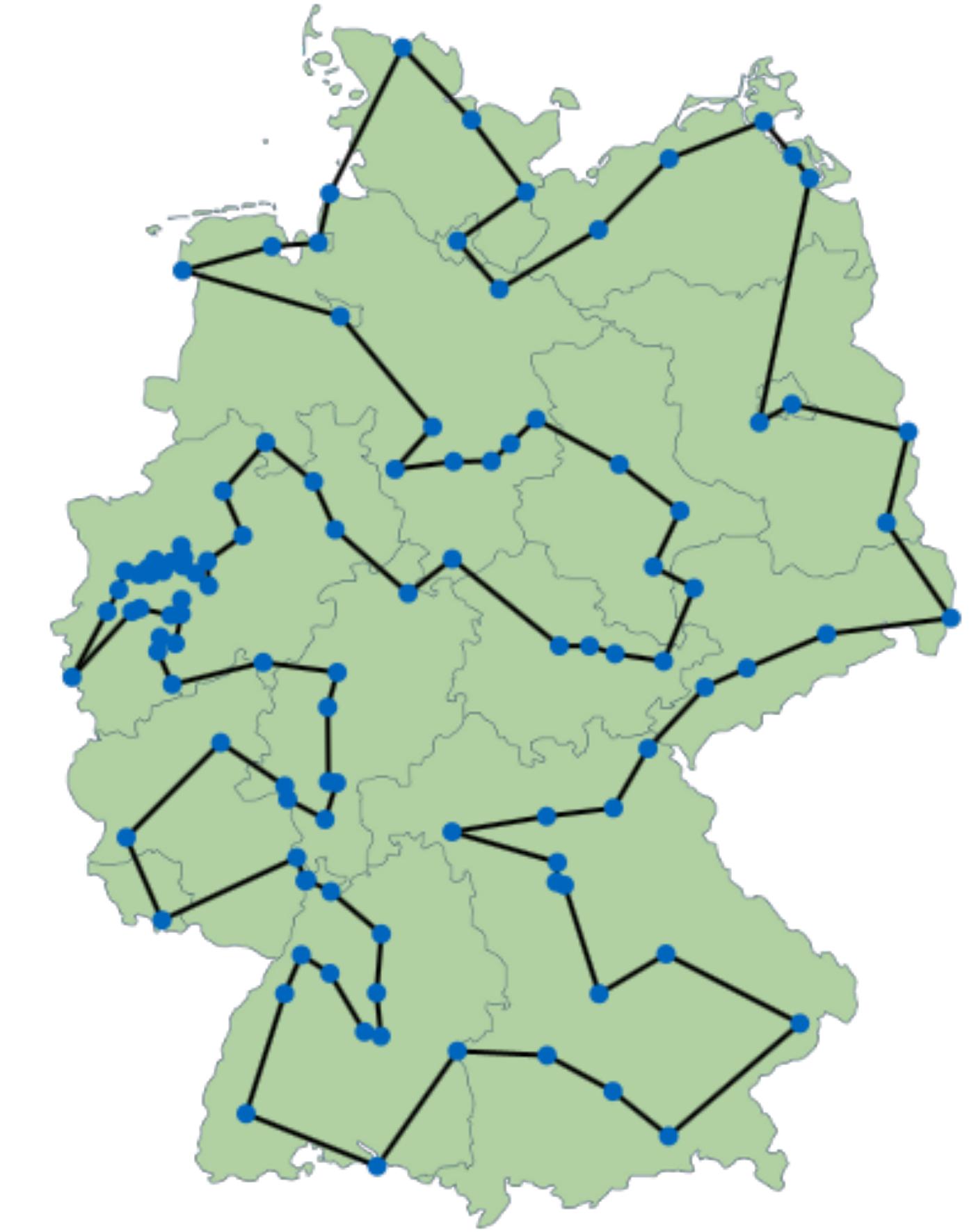
# Permutation problems

A solution is a permutation of objects

- Traveling Salesman Problem (TSP)
- For  $n$  cities there are  $n!$  possible routes
  - $53! \approx 4.27e69 > 1.2e68$

Number of atoms in the galaxy

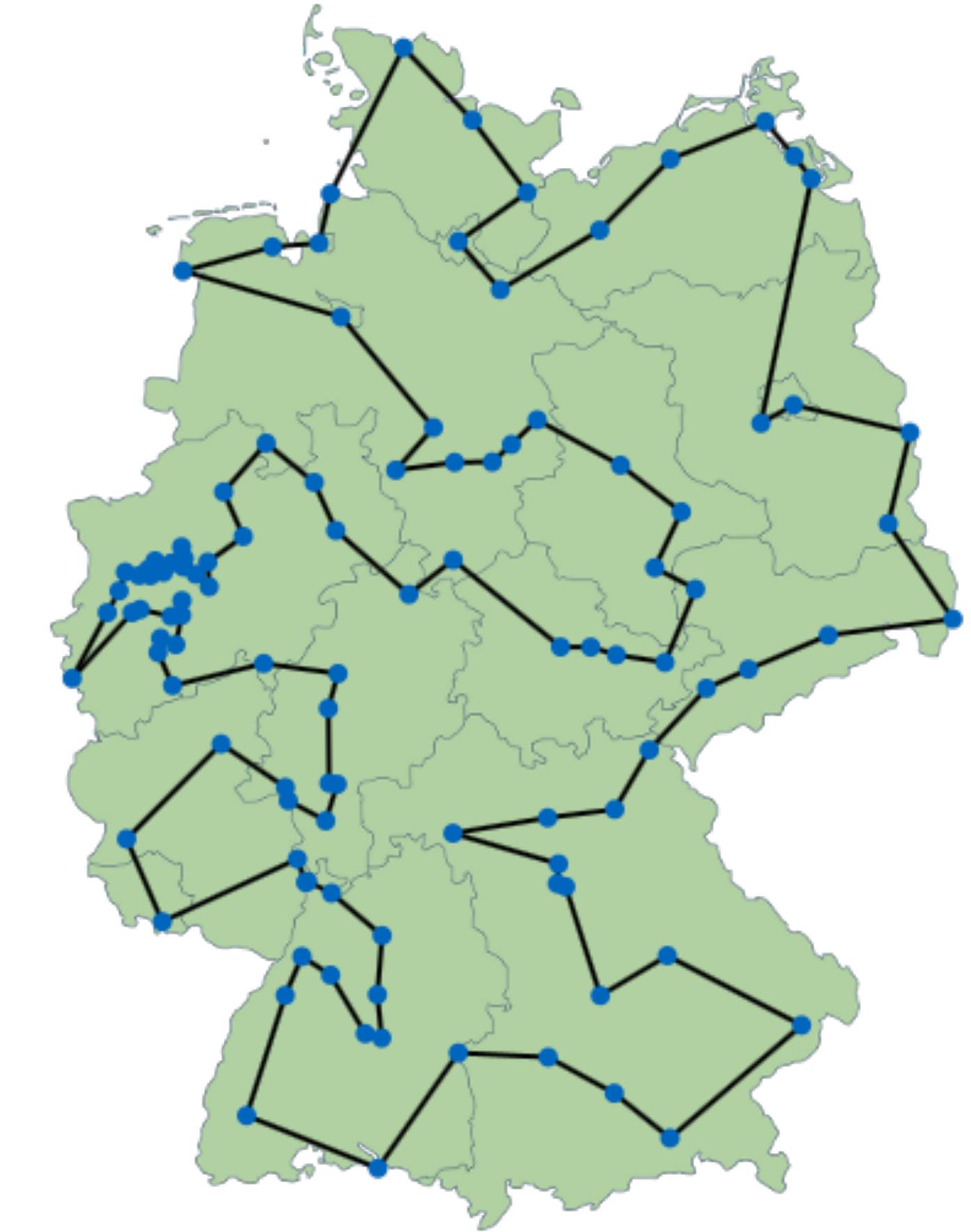
- For 15k towns it takes 22 years of computation to get the optimal tour
- Combinatorial optimization finds *good* solutions *quickly*
- Solution  $\sigma \in \mathbb{S}_n$ , Fitness function  $f: \mathbb{S}_n \rightarrow \mathbb{R}$ , instance, problem



# Permutation problems

A solution is a permutation of objects

- Traveling Salesman Problem (TSP)
- For  $n$  cities there are  $n!$  possible routes
  - $53! \approx 4.27e69 > 1.2e68$  Classical setting
  - Number of atoms in the galaxy
- For 15k towns it takes 22 years of computation to get the optimal tour
- Combinatorial optimization finds *good* solutions *quickly*
- Solution  $\sigma \in \mathbb{S}_n$ , Fitness function  $f: \mathbb{S}_n \rightarrow \mathbb{R}$ , instance, problem



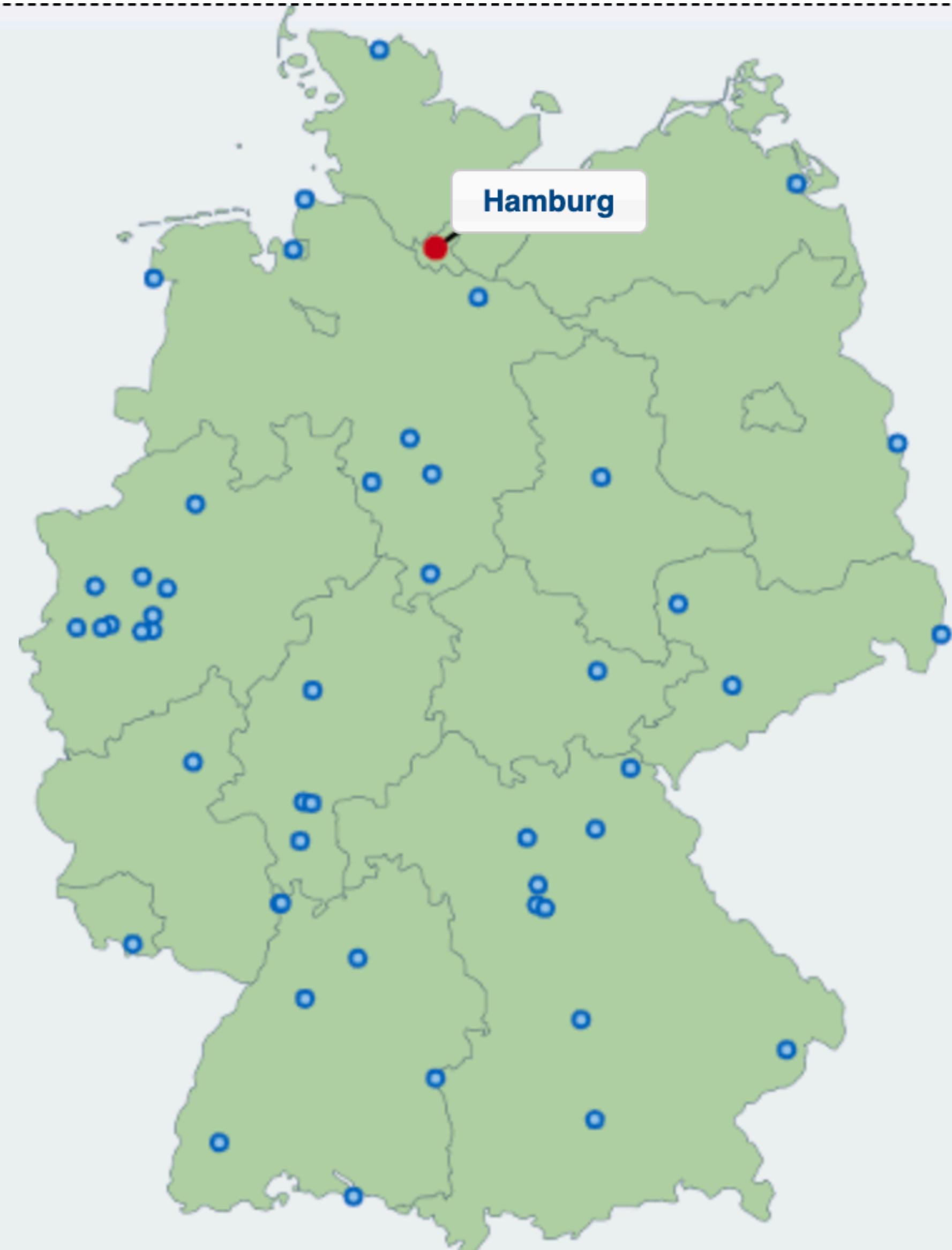
# Go to the closest city

## Classic greedy strategies



# Go to the closest city

## Classic greedy strategies



# Go to the closest city

## Classic greedy strategies



# Go to the closest city

## Classic greedy strategies



# Go to the closest city

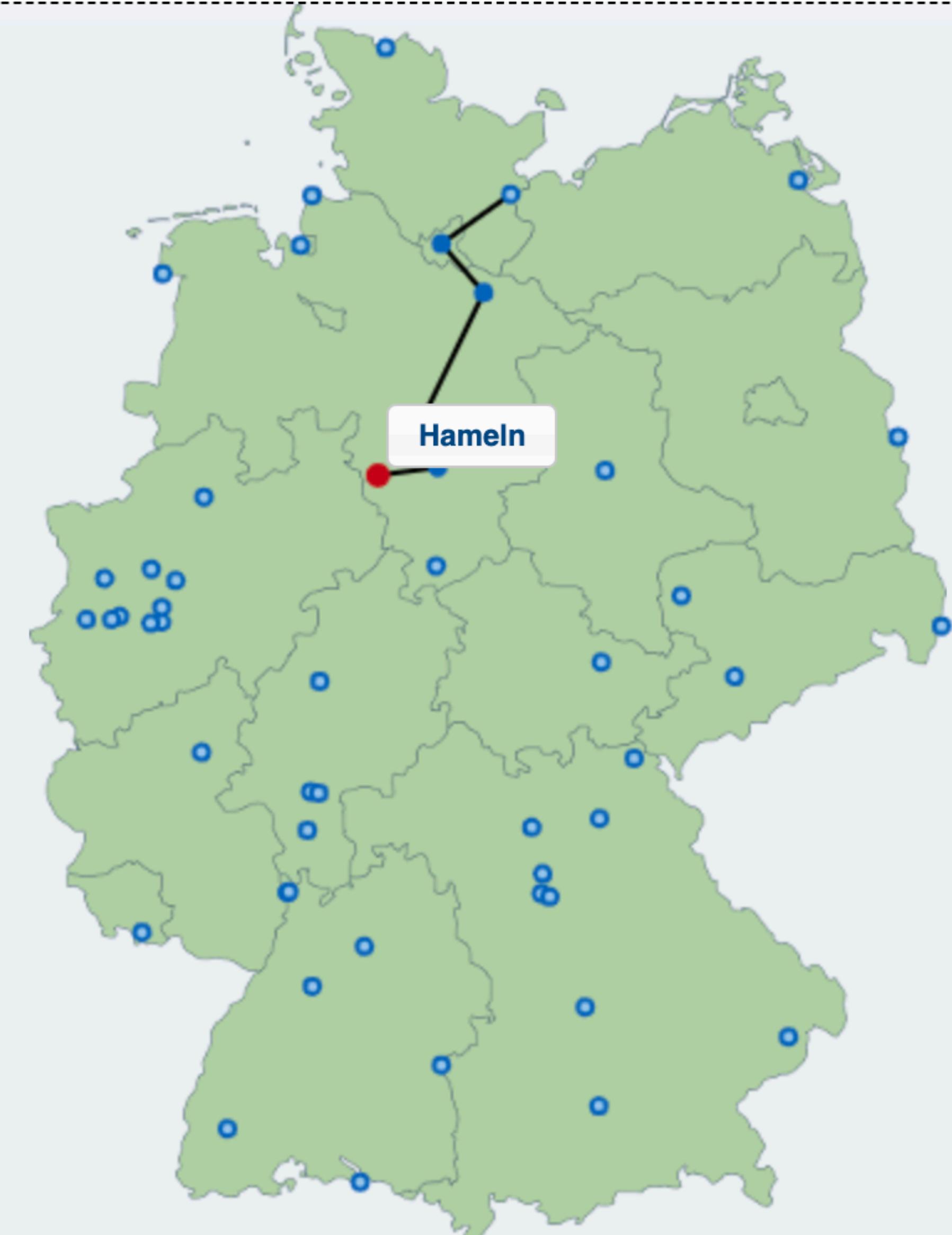
## Classic greedy strategies



# Go to the closest city

## Classic greedy strategies

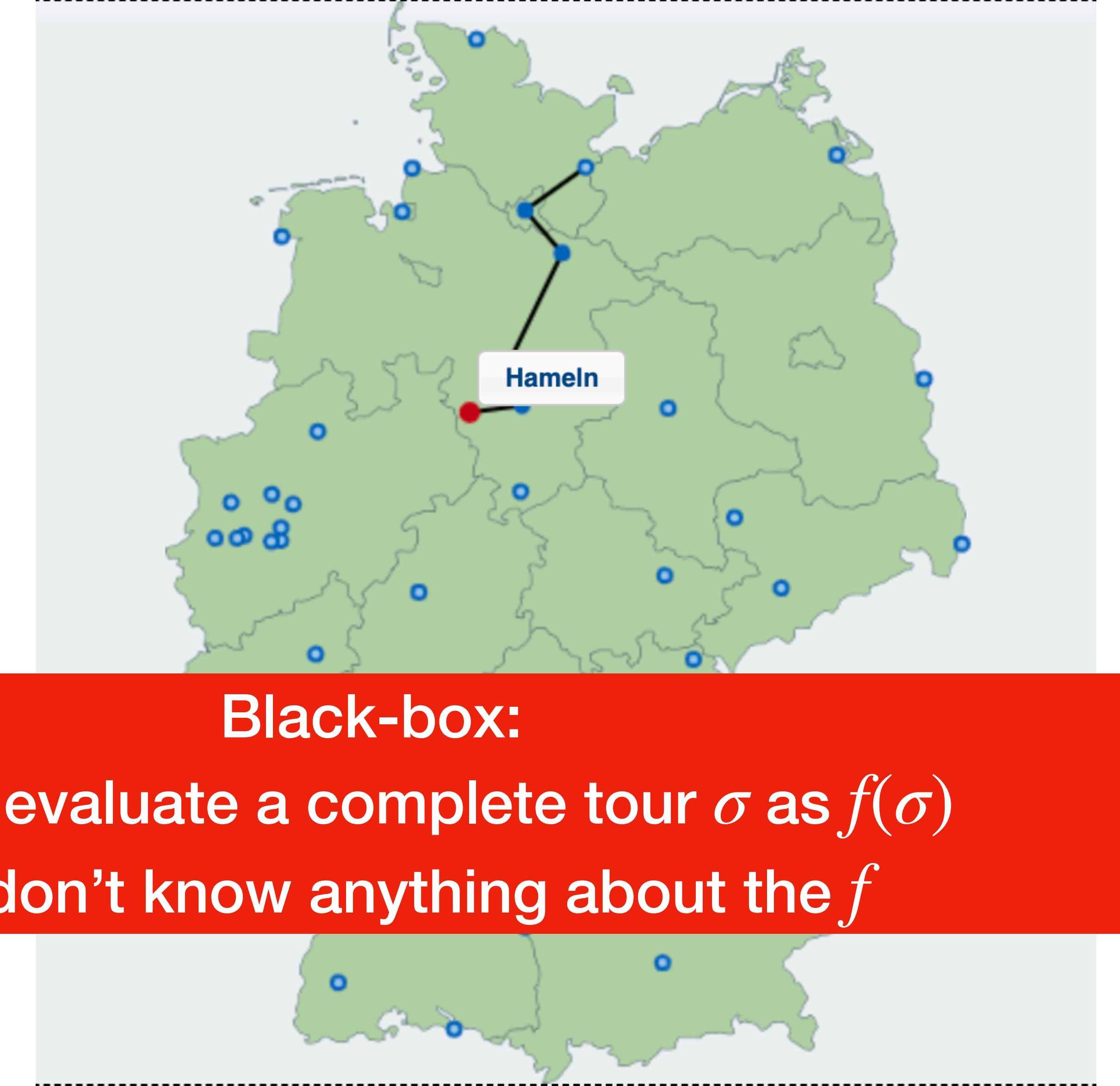
- No optimal but has good approximation guarantees
- Deterministic but there are probabilistic variants
- Easy to implement, easy to justify
- Your first option!
- **White-box:** We need to have access to the particular instance



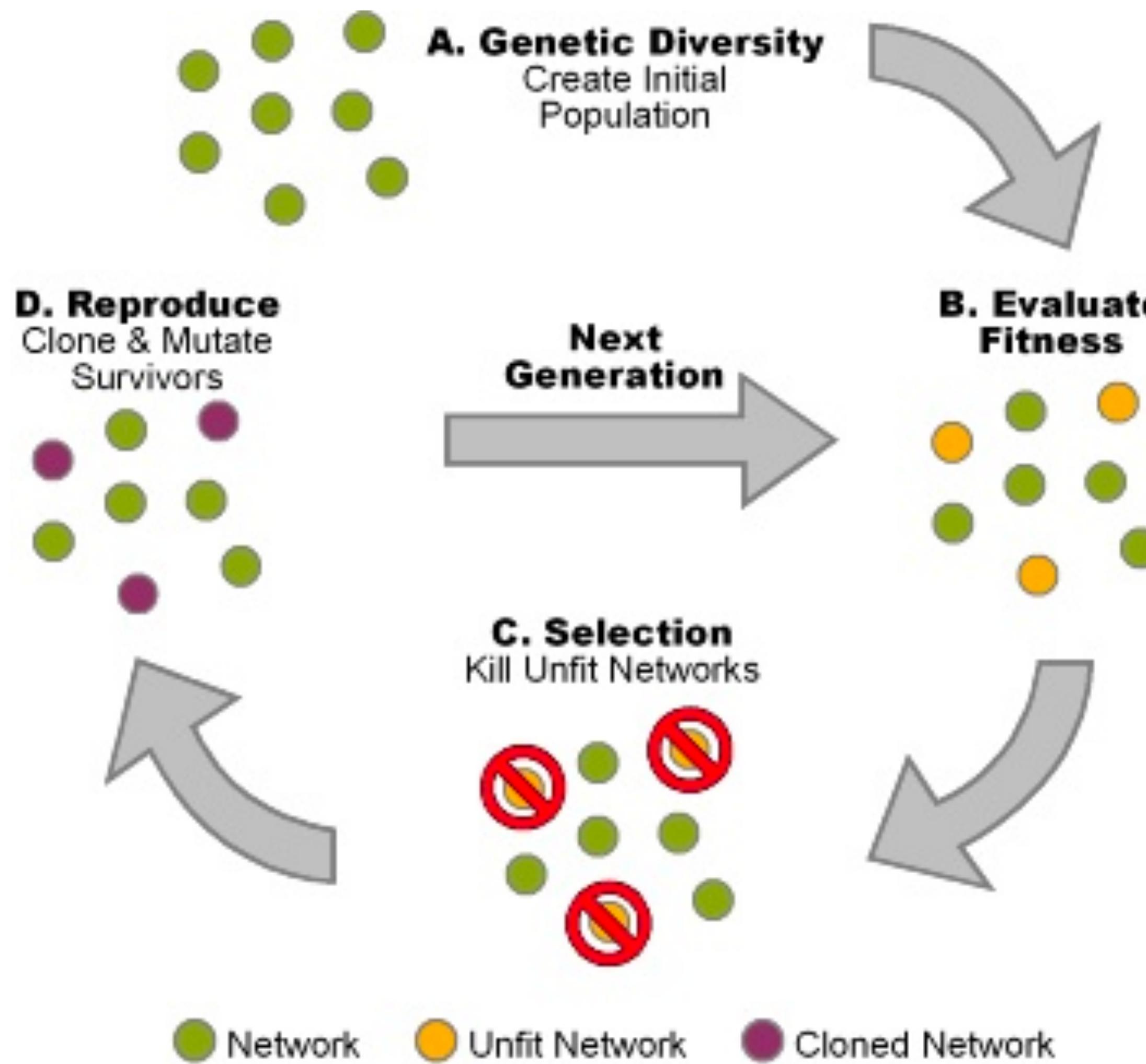
# Go to the closest city

## Classic greedy strategies

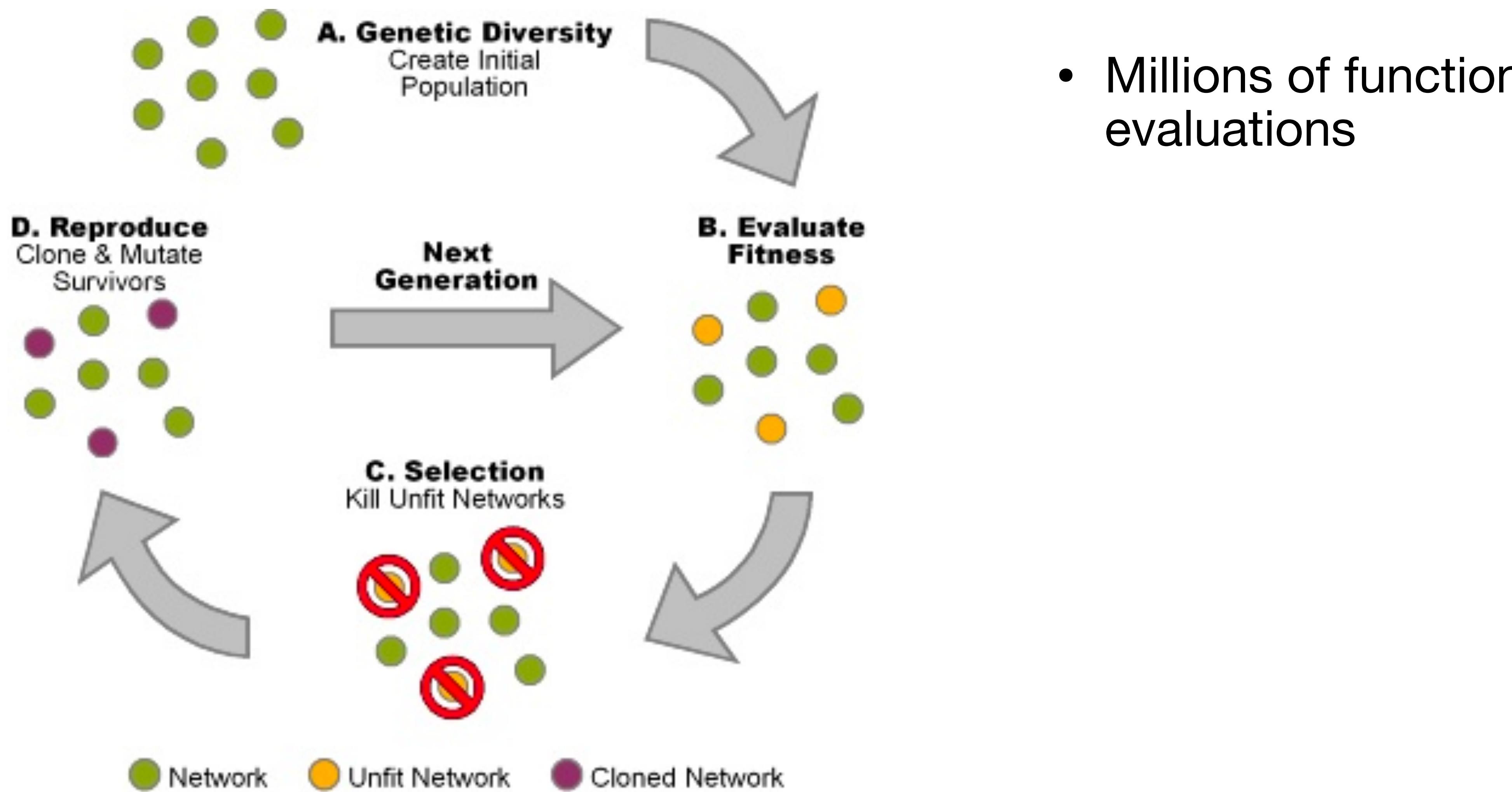
- No optimal tour has good approximation guarantees
- Deterministic but there are probabilistic variants
- Easy to implement, easy to analyze
- Your first option!
- **White-box:** We need to have access to the particular instance



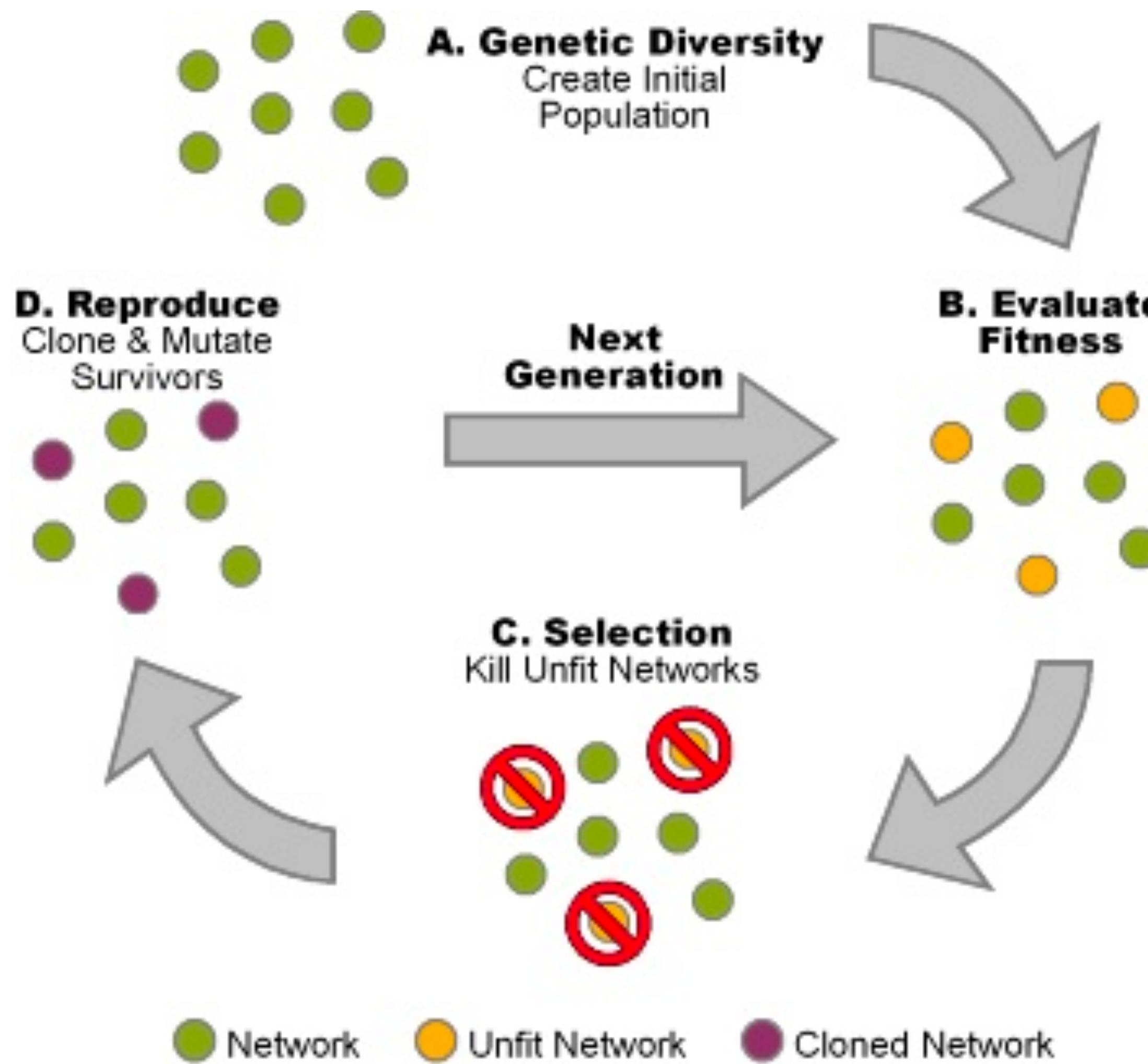
# Genetic algorithms are black-box optimizers



# Genetic algorithms are black-box optimizers

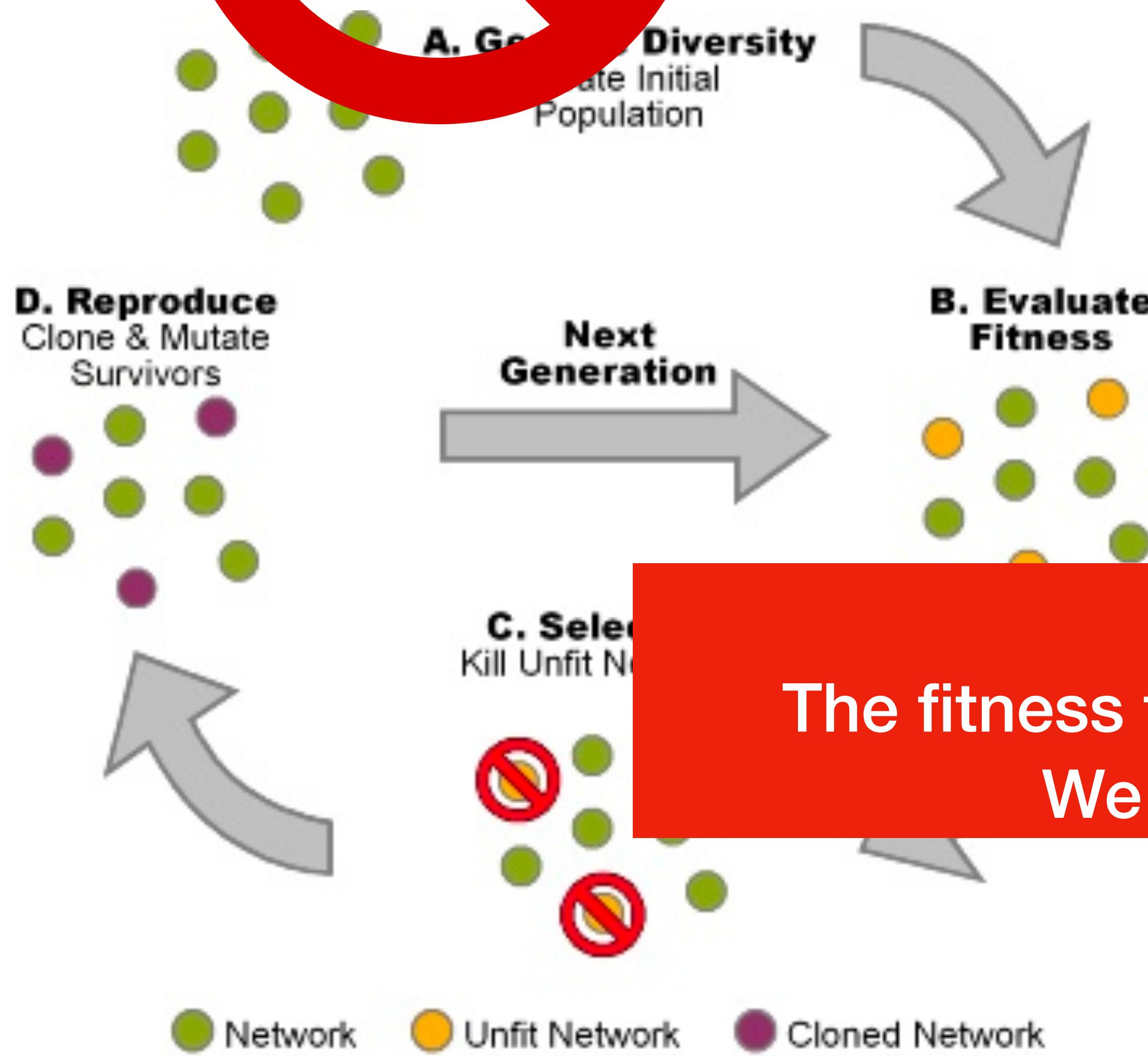


# Genetic algorithms are black-box optimizers



- Millions of function evaluations
- Based on having **cheap** evaluations of the objective function

# Genetic algorithms are black-box optimizers



- Millions of function evaluations
- Based on having **cheap** evaluations of the objective function

Expensive problem:  
The fitness function  $f$  takes lot of time/money  
We can evaluate it **few** times

# Problem setting

- Find the permutation  $\sigma$  that minimizes the fitness function  $f(\sigma)$  subject to
  - Black-box  $f$ : query the fitness function but not look at the instance
  - Expensive  $f$ : as few queries as possible

# Problem setting

- Find the permutation  $\sigma$  that minimizes the fitness function  $f(\sigma)$  subject to
  - Black-box  $f$ : query the fitness function but not look at the instance
  - Expensive  $f$ : as few queries as possible
- Bayesian optimization
  - CEGO is the state-of-the-art
  - Incremental sample
  - Gaussian process -> probability models for permutations

# Notation

- $\sigma = (3,1,2,5,4)$
- Bijection of the set  $[n]$  onto itself,  
Symmetric group,  $\mathbb{S}_n$
- $e = (1,2,3,4,5)$
- Inverse  $\sigma^{-1} : \sigma(i) = j \Leftrightarrow \sigma^{-1}(j) = i$
- Composition,  $\sigma \cdot \pi(i) = \sigma(\pi(i))$

# Notation

- $\sigma = (3,1,2,5,4)$
  - Bijection of the set  $[n]$  onto itself,  
Symmetric group,  $\mathbb{S}_n$
  - $e = (1,2,3,4,5)$
  - Inverse  $\sigma^{-1} : \sigma(i) = j \Leftrightarrow \sigma^{-1}(j) = i$
  - Composition,  $\sigma \cdot \pi(i) = \sigma(\pi(i))$
- Interpretation
    - Ranking  $\sigma = (3,5,2,1,4)$
    - Ordering  $\sigma^{-1} = (4,3,1,5,2)$

# UMM, Unbalanced Mallows model

## Big picture of our proposal

$S \leftarrow 10$  random permutations

for  $i \in [1\dots evaluations]$  :

$$\forall \sigma \in S : w_\sigma \leftarrow \rho^{f(\sigma)}$$

$$\mu \leftarrow wmedian(S, \{w_\sigma\})$$

$$\sigma' \sim M(\mu, \theta_i)$$

$$S \leftarrow S \cup \{\sigma'\}$$

# UMM, Unbalanced Mallows model

## Big picture of our proposal

$S \leftarrow 10$  random permutations

for  $i \in [1\dots evaluations]$  :

$$\forall \sigma \in S : w_\sigma \leftarrow \rho^{f(\sigma)}$$

$$\mu \leftarrow wmedian(S, \{w_\sigma\})$$

$$\sigma' \sim M(\mu, \theta_i)$$

$$S \leftarrow S \cup \{\sigma'\}$$

$$w_\sigma = \begin{cases} 0 & \arg \max_{\sigma \in S} f(\sigma) \\ 1 & \arg \min_{\sigma \in S} f(\sigma) \end{cases}$$

# UMM, Unbalanced Mallows model

## Big picture of our proposal

$S \leftarrow 10$  random permutations

for  $i \in [1\dots evaluations]$  :

$$\forall \sigma \in S : w_\sigma \leftarrow \rho^{f(\sigma)}$$

$$\mu \leftarrow wmedian(S, \{w_\sigma\})$$

$$\sigma' \sim M(\mu, \theta_i)$$

$$S \leftarrow S \cup \{\sigma'\}$$

$$w_\sigma = \begin{cases} 0 & \arg \max_{\sigma \in S} f(\sigma) \\ 1 & \arg \min_{\sigma \in S} f(\sigma) \end{cases}$$

$$\arg \min_{\sigma \in \mathbb{S}_n} \sum_{\sigma_i \in S} d(\sigma, \sigma_i)$$

**Median**, the permutation  $\sigma \in S_n$  *in the middle* of  $S$

# **Median, the permutation $\sigma \in S_n$ *in the middle* of $S$**

- $\mu = \arg \min_{\sigma \in \mathbb{S}_n} \sum_{\sigma_i \in S} d(\sigma, \sigma_i)$

# Median, the permutation $\sigma \in S_n$ *in the middle* of $S$

- $\mu = \arg \min_{\sigma \in \mathbb{S}_n} \sum_{\sigma_i \in S} d(\sigma, \sigma_i)$
- Kendall's- $\tau$  distance,  $d(\sigma, \sigma') = \sum_{i < j} \mathbb{I}\{(\sigma(i) - \sigma(j))(\sigma'(i) - \sigma'(j)) < 0\}$

# Median, the permutation $\sigma \in S_n$ *in the middle* of $S$

- $\mu = \arg \min_{\sigma \in \mathbb{S}_n} \sum_{\sigma_i \in S} d(\sigma, \sigma_i)$
- Kendall's- $\tau$  distance,  $d(\sigma, \sigma') = \sum_{i < j} \mathbb{I}\{(\sigma(i) - \sigma(j))(\sigma'(i) - \sigma'(j)) < 0\}$
- Kemeny is NP-hard

# Median, the permutation $\sigma \in S_n$ in the middle of S

- $\mu = \arg \min_{\sigma \in \mathbb{S}_n} \sum_{\sigma_i \in S} d(\sigma, \sigma_i)$
- Kendall's- $\tau$  distance,  $d(\sigma, \sigma') = \sum_{i < j} \mathbb{I}\{(\sigma(i) - \sigma(j))(\sigma'(i) - \sigma'(j)) < 0\}$
- Kemeny is NP-hard

When solving a problem of interest,  
do not solve a more general problem  
as an intermediate step

# Borda is a *good* approximation of the median

	Item 1	Item 2	Item 3	Item 4
$\sigma_1$	1	2	4	3
$\sigma_2$	4	2	3	1
$\sigma_3$	1	4	2	3
$\sigma_4$	2	1	4	3

# Borda is a good approximation of the median

	Item 1	Item 2	Item 3	Item 4
$\sigma_1$	1	2	4	3
$\sigma_2$	4	2	3	1
$\sigma_3$	1	4	2	3
$\sigma_4$	2	1	4	3

8	9	13	10
---	---	----	----

- For each item  $i$ , the Borda score is  
$$B(i) = \sum_{\sigma} \sigma(i)$$

# Borda is a good approximation of the median

	Item 1	Item 2	Item 3	Item 4
$\sigma_1$	1	2	4	3
$\sigma_2$	4	2	3	1
$\sigma_3$	1	4	2	3
$\sigma_4$	2	1	4	3

- For each item  $i$ , the Borda score is  
$$B(i) = \sum_{\sigma} \sigma(i)$$

8	9	13	10
---	---	----	----

1	2	4	3
---	---	---	---

# Borda is a good approximation of the median

	Item 1	Item 2	Item 3	Item 4
$\sigma_1$	1	2	4	3
$\sigma_2$	4	2	3	1
$\sigma_3$	1	4	2	3
$\sigma_4$	2	1	4	3

- For each item  $i$ , the Borda score is  $B(i) = \sum_{\sigma} \sigma(i)$
- Makes sense for rankings

8	9	13	10
---	---	----	----

1	2	4	3
---	---	---	---

# Borda is a good approximation of the median

	Item 1	Item 2	Item 3	Item 4
$\sigma_1$	1	2	4	3
$\sigma_2$	4	2	3	1
$\sigma_3$	1	4	2	3
$\sigma_4$	2	1	4	3

8	9	13	10
---	---	----	----

1	2	4	3
---	---	---	---

- For each item  $i$ , the Borda score is  $B(i) = \sum_{\sigma} \sigma(i)$
- Makes sense for rankings
- Computationally efficient, complexity comes from sorting

# Borda is a good approximation of the median

	Item 1	Item 2	Item 3	Item 4
$\sigma_1$	1	2	4	3
$\sigma_2$	4	2	3	1
$\sigma_3$	1	4	2	3
$\sigma_4$	2	1	4	3

8	9	13	10
---	---	----	----

1	2	4	3
---	---	---	---

- For each item  $i$ , the Borda score is  $B(i) = \sum_{\sigma} \sigma(i)$
- Makes sense for rankings
- Computationally efficient, complexity comes from sorting
- It is a 5/11 approximation

# Borda is a good approximation of the median

	Item 1	Item 2	Item 3	Item 4
$\sigma_1$	1	2	4	3
$\sigma_2$	4	2	3	1
$\sigma_3$	1	4	2	3
$\sigma_4$	2	1	4	3

8	9	13	10
---	---	----	----

1	2	4	3
---	---	---	---

- For each item  $i$ , the Borda score is  $B(i) = \sum_{\sigma} \sigma(i)$
- Makes sense for rankings
- Computationally efficient, complexity comes from sorting
- It is a 5/11 approximation
- Unbiased estimator

# Weighted Borda

	<b>Item 1</b>	<b>Item 2</b>	<b>Item 3</b>	<b>Item 4</b>	<b>W</b>
$\sigma_1$	1	2	4	3	1
$\sigma_2$	4	2	3	1	3
$\sigma_3$	1	4	2	3	1
$\sigma_4$	2	1	4	3	2

# Weighted Borda

	Item 1	Item 2	Item 3	Item 4	
$\sigma_1$	1	2	4	3	W
$\sigma_2$	4	2	3	1	1
$\sigma_3$	1	4	2	3	3
$\sigma_4$	2	1	4	3	1
					2

- For each item  $i$ , the weighted Borda score is  
$$B(i) = \sum_{\sigma} \sigma(i) w(\sigma)$$

18	14	23	17
----	----	----	----

# Weighted Borda

	Item 1	Item 2	Item 3	Item 4	
$\sigma_1$	1	2	4	3	W
$\sigma_2$	4	2	3	1	1
$\sigma_3$	1	4	2	3	3
$\sigma_4$	2	1	4	3	1
					2

- For each item  $i$ , the weighted Borda score is  
$$B(i) = \sum_{\sigma} \sigma(i) w(\sigma)$$

18	14	23	17
----	----	----	----

3	1	4	2
---	---	---	---

# Weighted Borda

	Item 1	Item 2	Item 3	Item 4	
$\sigma_1$	1	2	4	3	W
$\sigma_2$	4	2	3	1	1
$\sigma_3$	1	4	2	3	3
$\sigma_4$	2	1	4	3	1
					2

- For each item  $i$ , the weighted Borda score is  $B(i) = \sum_{\sigma} \sigma(i) w(\sigma)$
- Equivalent to replicating the permutations proportionally to their weight

18	14	23	17
----	----	----	----

3	1	4	2
---	---	---	---

# Weighted Borda

	Item 1	Item 2	Item 3	Item 4	
$\sigma_1$	1	2	4	3	W
$\sigma_2$	4	2	3	1	1
$\sigma_3$	1	4	2	3	3
$\sigma_4$	2	1	4	3	1
	18	14	23	17	2

- For each item  $i$ , the weighted Borda score is  $B(i) = \sum_{\sigma} \sigma(i) w(\sigma)$
- Equivalent to replicating the permutations proportionally to their weight
- The complexity is the same

3	1	4	2
---	---	---	---

# Weighted Borda

	Item 1	Item 2	Item 3	Item 4	
$\sigma_1$	1	2	4	3	W
$\sigma_2$	4	2	3	1	1
$\sigma_3$	1	4	2	3	3
$\sigma_4$	2	1	4	3	1
	18	14	23	17	2

- For each item  $i$ , the weighted Borda score is  $B(i) = \sum_{\sigma} \sigma(i) w(\sigma)$
- Equivalent to replicating the permutations proportionally to their weight
- The complexity is the same
- It can be shown that provided certain conditions, it is

# Weighted Borda

	Item 1	Item 2	Item 3	Item 4	
$\sigma_1$	1	2	4	3	W
$\sigma_2$	4	2	3	1	1
$\sigma_3$	1	4	2	3	3
$\sigma_4$	2	1	4	3	1
	18	14	23	17	2

- For each item  $i$ , the weighted Borda score is  $B(i) = \sum_{\sigma} \sigma(i) w(\sigma)$
- Equivalent to replicating the permutations proportionally to their weight
- The complexity is the same
- It can be shown that provided certain conditions, it is
  - More robust estimator [1,2]

# Weighted Borda

	Item 1	Item 2	Item 3	Item 4	
$\sigma_1$	1	2	4	3	W
$\sigma_2$	4	2	3	1	1
$\sigma_3$	1	4	2	3	3
$\sigma_4$	2	1	4	3	1
	18	14	23	17	2

- For each item  $i$ , the weighted Borda score is  $B(i) = \sum_{\sigma} \sigma(i) w(\sigma)$
- Equivalent to replicating the permutations proportionally to their weight
- The complexity is the same
- It can be shown that provided certain conditions, it is
  - More robust estimator [1,2]
  - Unbiased estimator [3]

# UMM, Unbalanced Mallows model

$S \leftarrow 10$  random permutations

for  $i \in [1\dots evaluations]$  :

$$\forall \sigma \in S : w_\sigma \leftarrow \rho^{f(\sigma)}$$

$$\mu \leftarrow wmedian(S, \{w_\sigma\})$$

$$\sigma' \sim M(\mu, \theta_i)$$

$$S \leftarrow S \cup \{\sigma'\}$$

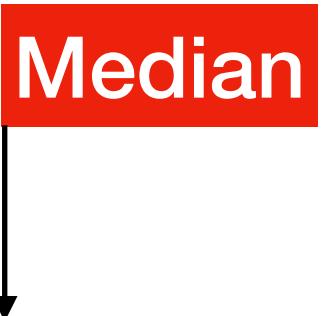
# Mallows, probability distribution for permutation

Analogous to the Gaussian distribution

- $p(\sigma | \mu, \theta) \propto \exp(-\theta d(\sigma, \mu))$
- Easy to sample [1,4]
- For  $S \sim M(\mu, \theta)$ , Kemeny is the MLE and Borda is the consistent estimator

# Mallows, probability distribution for permutation

Analogous to the Gaussian distribution



- $p(\sigma | \mu, \theta) \propto \exp(-\theta d(\sigma, \mu))$
- Easy to sample [1,4]
- For  $S \sim M(\mu, \theta)$ , Kemeny is the MLE and Borda is the consistent estimator

# Mallows, probability distribution for permutation

Analogous to the Gaussian distribution

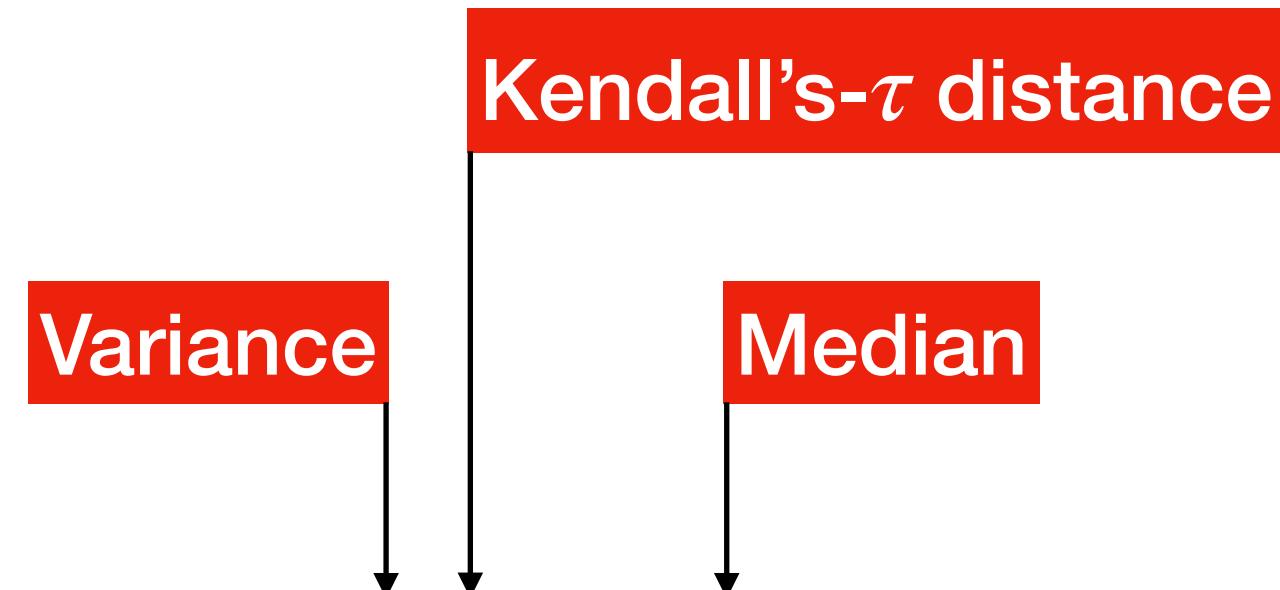
Kendall's- $\tau$  distance

Median

- $p(\sigma | \mu, \theta) \propto \exp(-\theta d(\sigma, \mu))$
- Easy to sample [1,4]
- For  $S \sim M(\mu, \theta)$ , Kemeny is the MLE and Borda is the consistent estimator

# Mallows, probability distribution for permutation

Analogous to the Gaussian distribution



- $p(\sigma | \mu, \theta) \propto \exp(-\theta d(\sigma, \mu))$
- Easy to sample [1,4]
- For  $S \sim M(\mu, \theta)$ , Kemeny is the MLE and Borda is the consistent estimator

# The uncertainty of the median decreases with iterations

## Decreasing variance

- The variance is not intuitive
- For  $\sigma \sim M(\mu, \theta)$ 
  - $\mathbb{E}[d(\sigma, \mu)] = \frac{n \cdot \exp(-\theta)}{1 - \exp(-\theta)} - \sum_{j=1}^n \frac{j \cdot \exp(-j\theta)}{1 - \exp(-j\theta)}$
  - $\mathbb{E}[d(\sigma, \mu)]$  is a function of  $\theta$
  - The  $\mathbb{E}[d(\sigma, \mu)]$  linearly decreases with iterations

# Exploration - exploitation

The  $\mathbb{E}[d(\mu, \sigma)]$  linearly decreases with the iterations

$S \leftarrow 10$  random permutations

for  $i \in [1 \dots evaluations]$  :

$$\forall \sigma \in S : w_\sigma \leftarrow \rho^{f(\sigma)}$$

The sample is more accurate

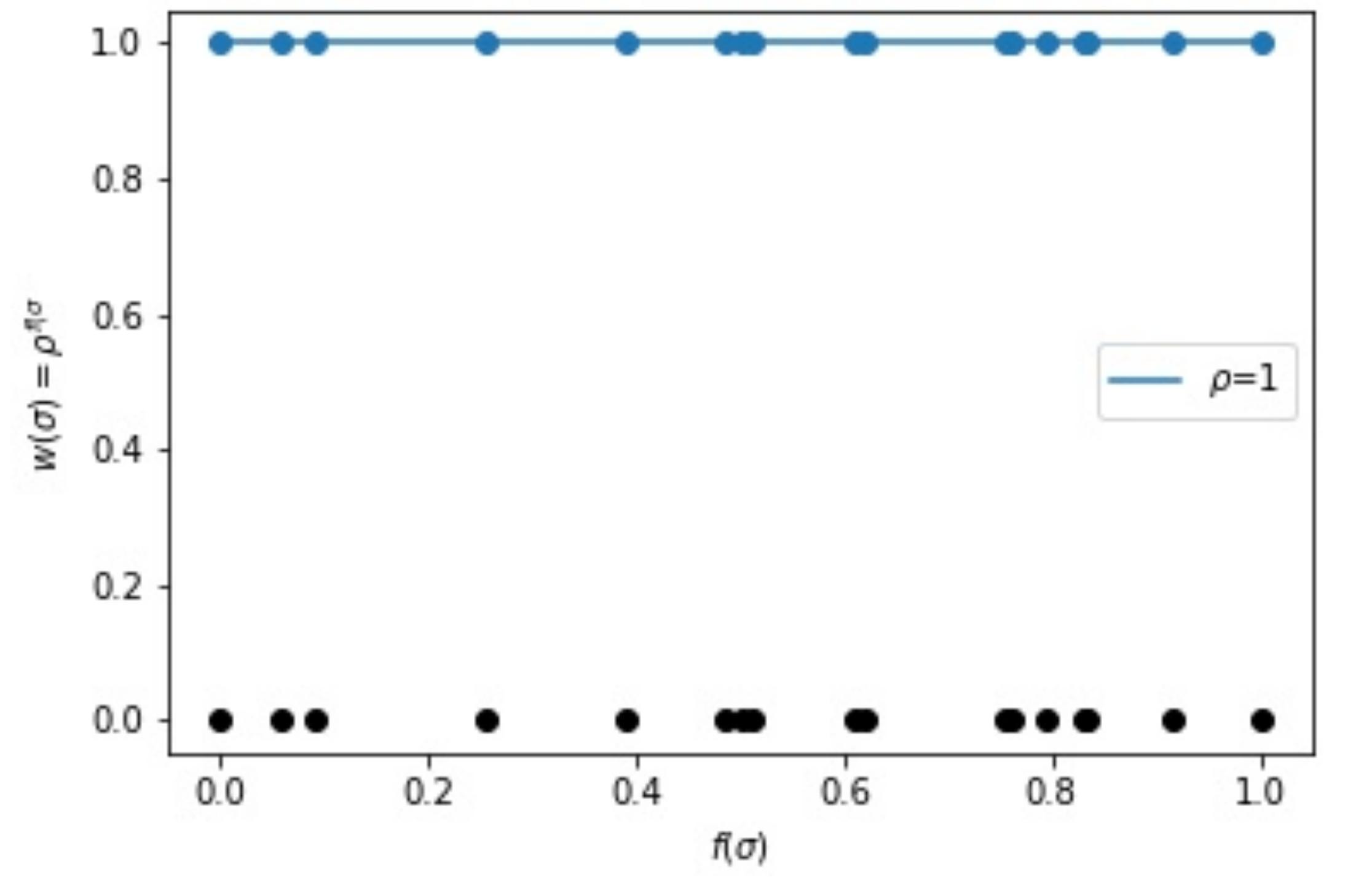
$$\mu \leftarrow wmedian(S, \{w_\sigma\})$$

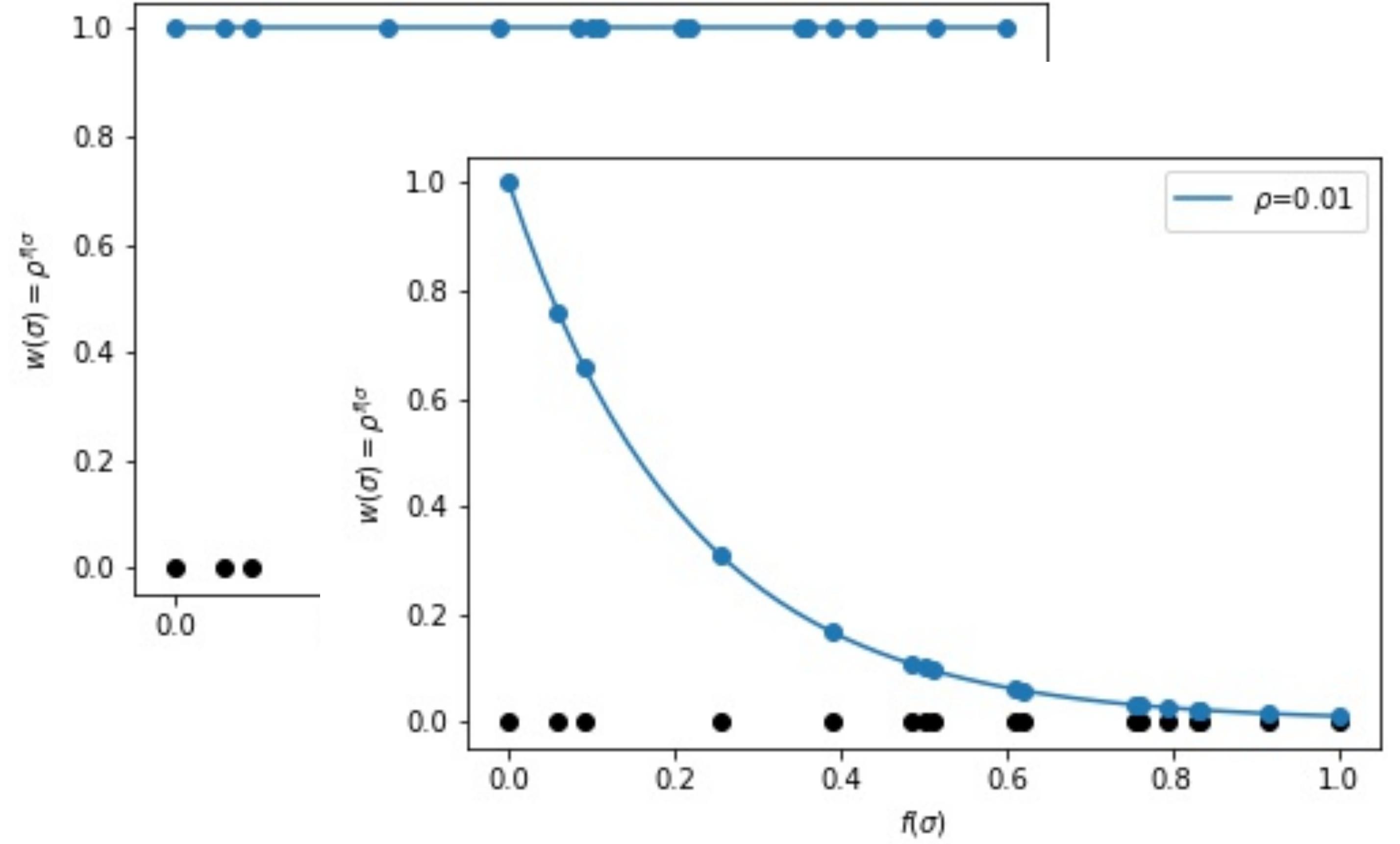
$$\sigma' \sim M(\mu, \theta_i)$$

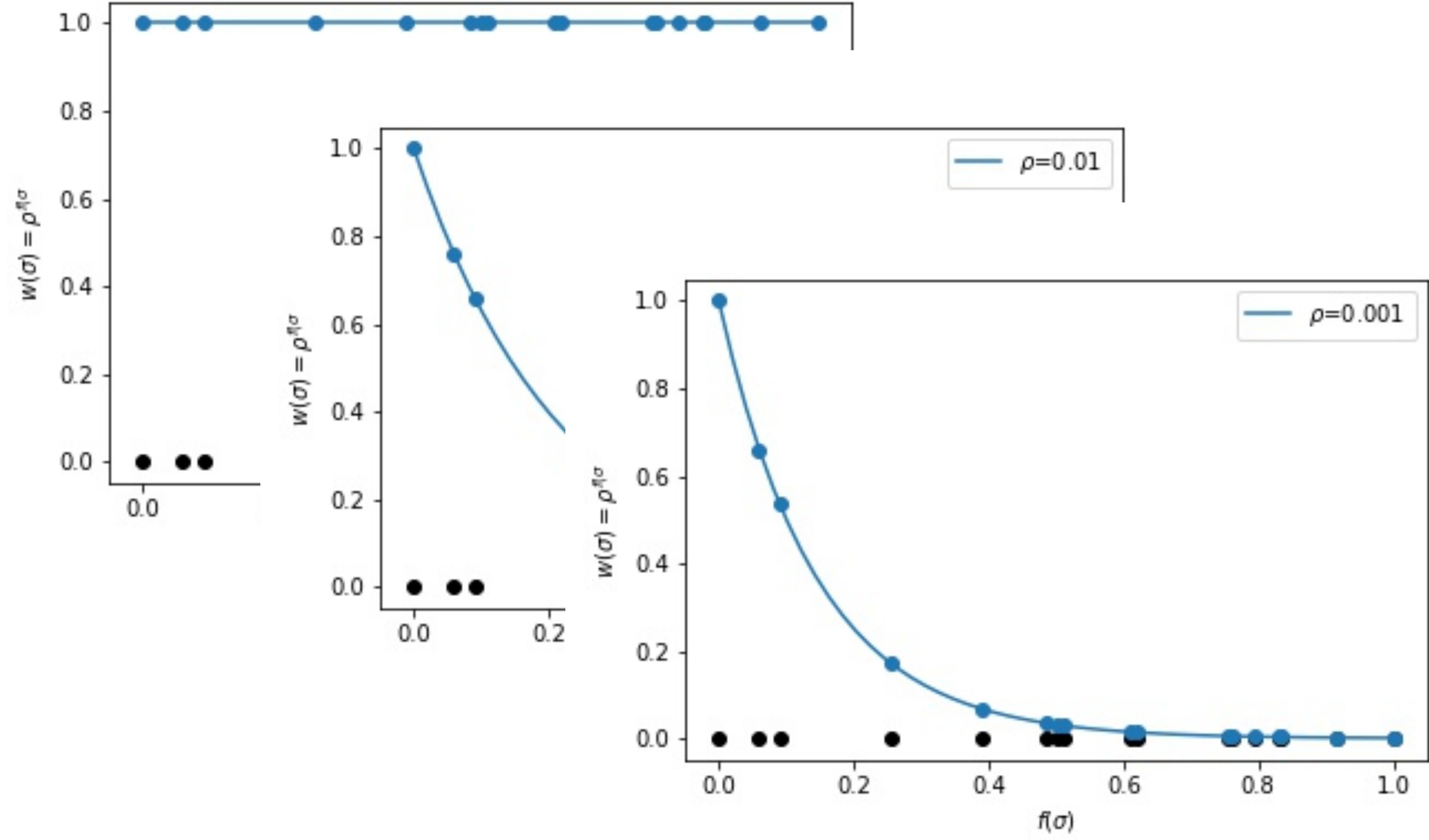
The estimator is more accurate

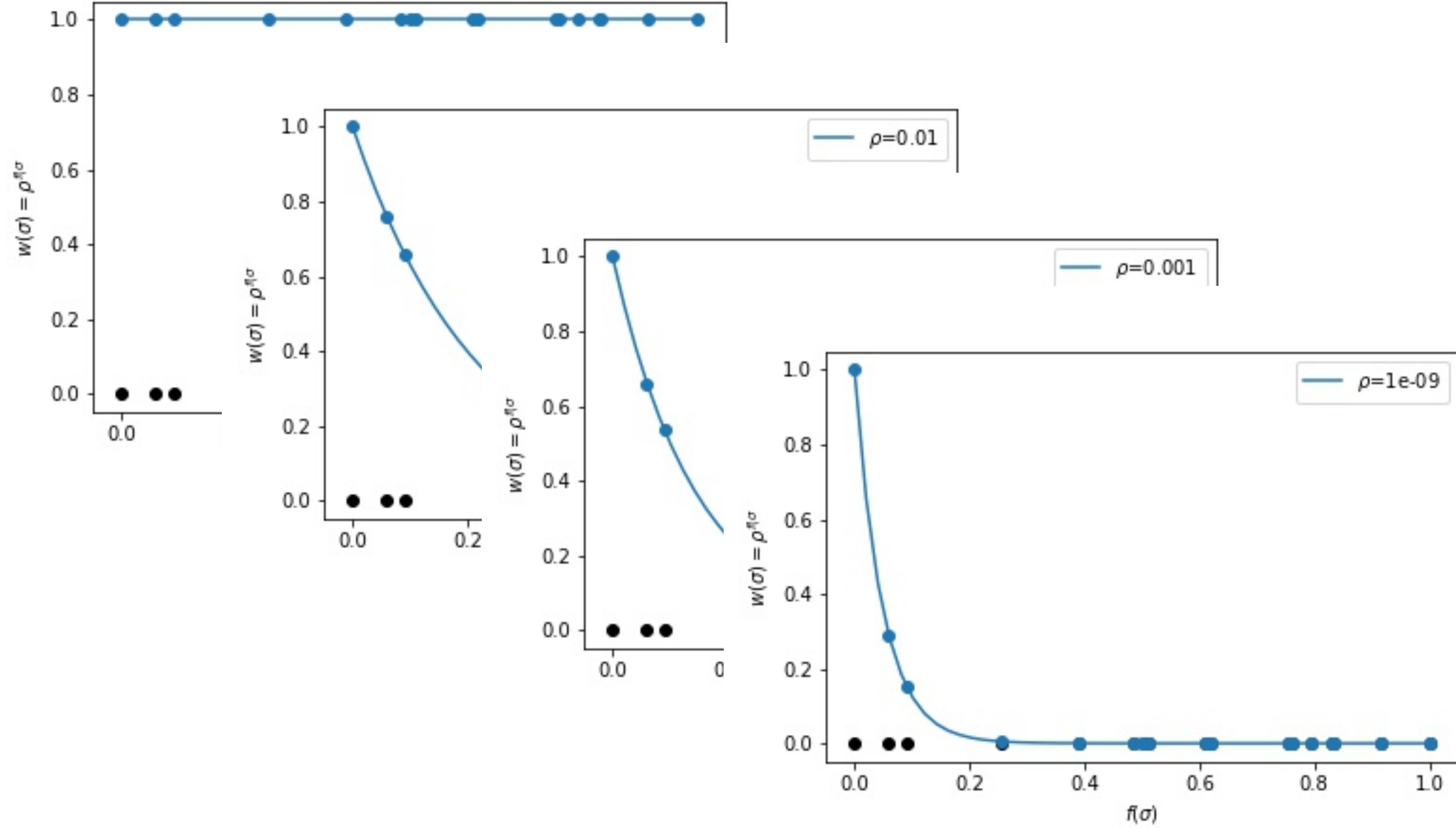
$$S \leftarrow S \cup \{\sigma'\}$$

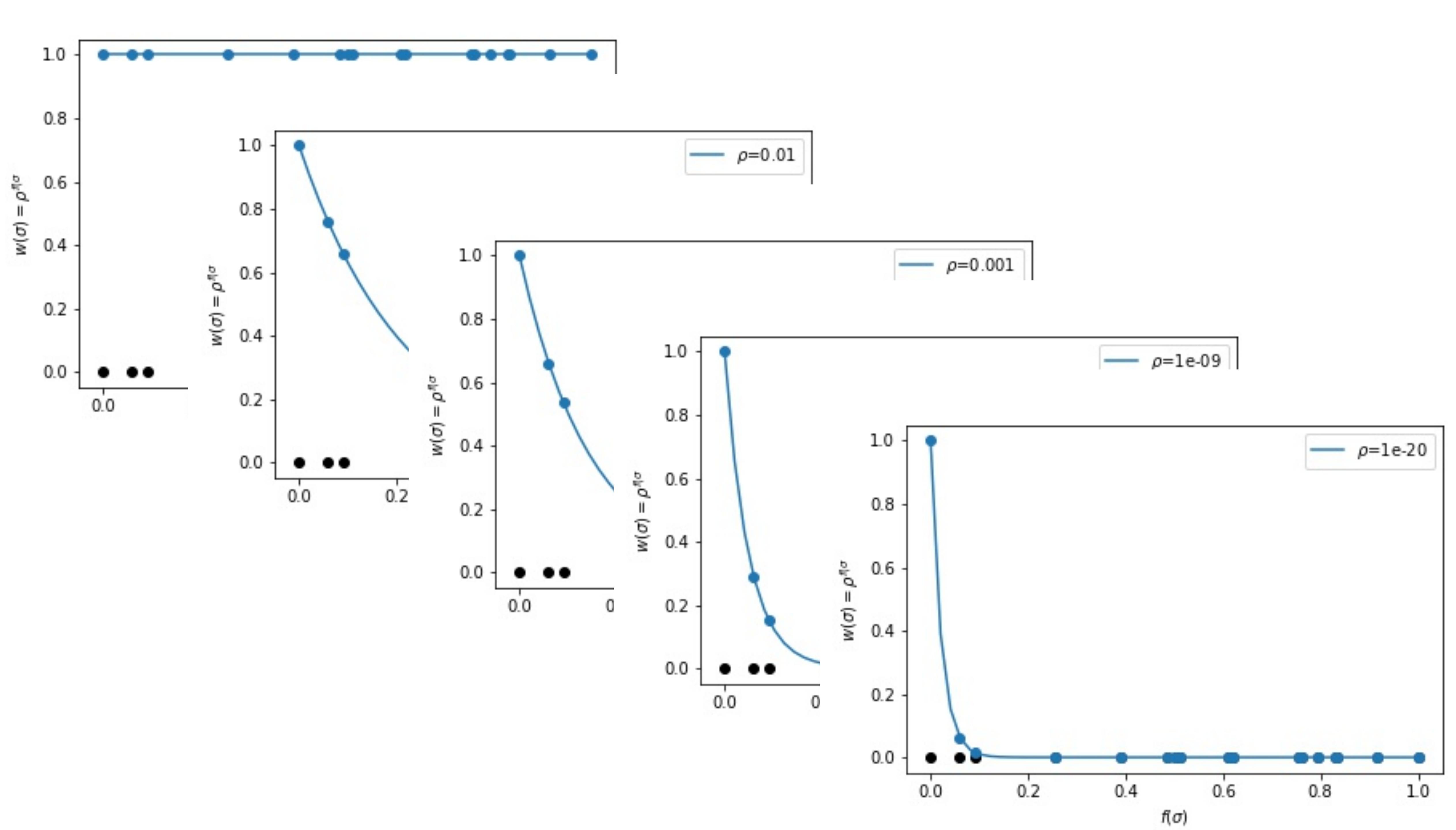






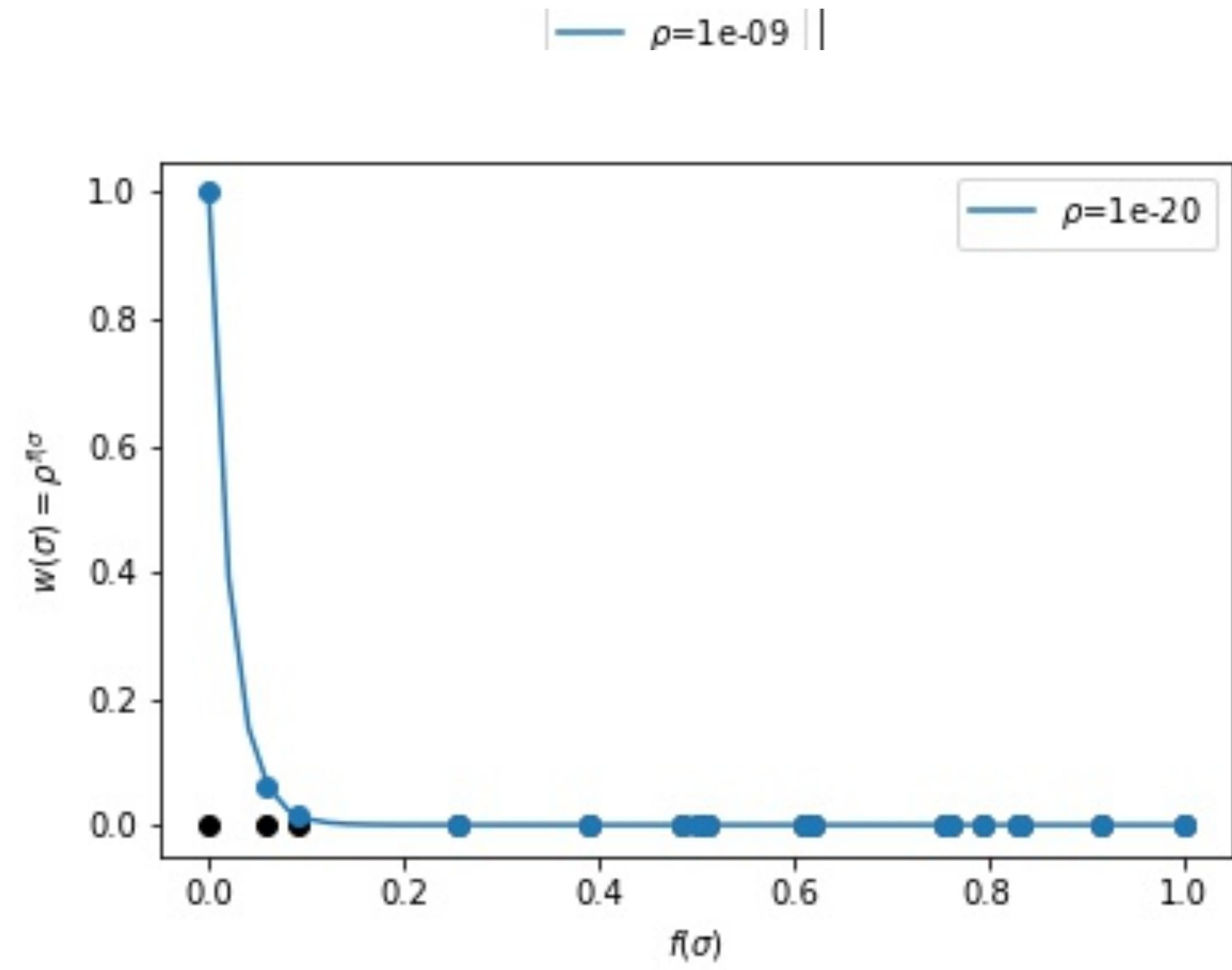
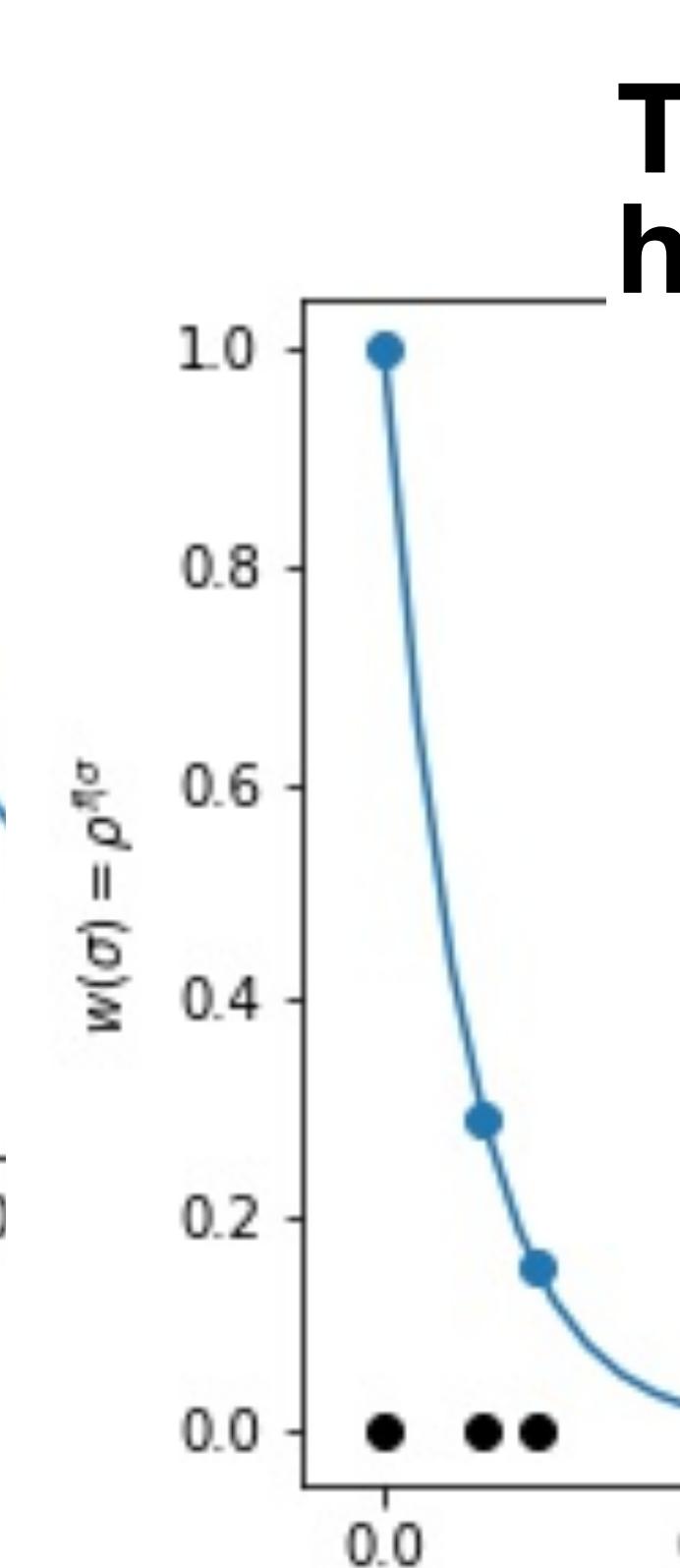
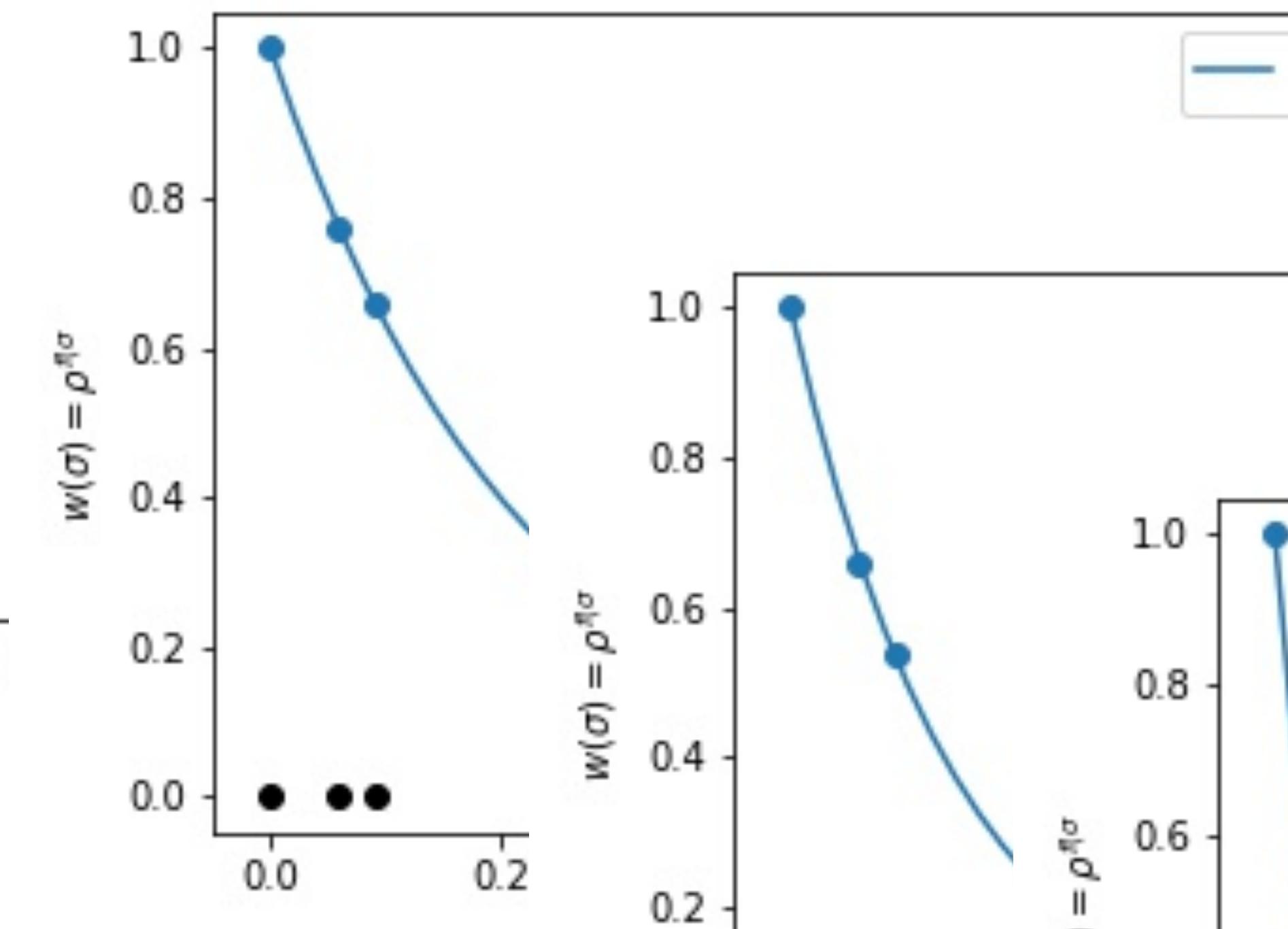
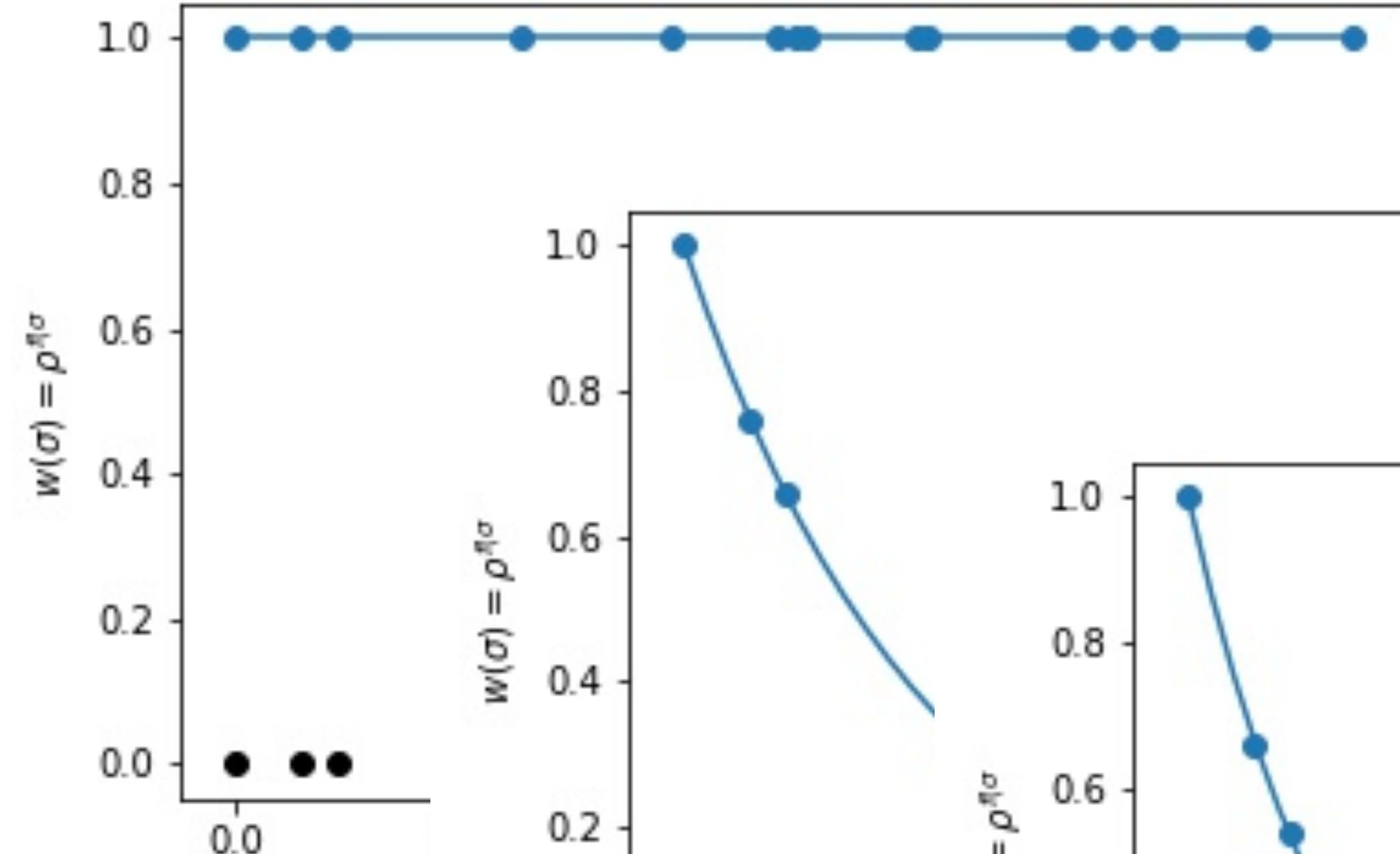






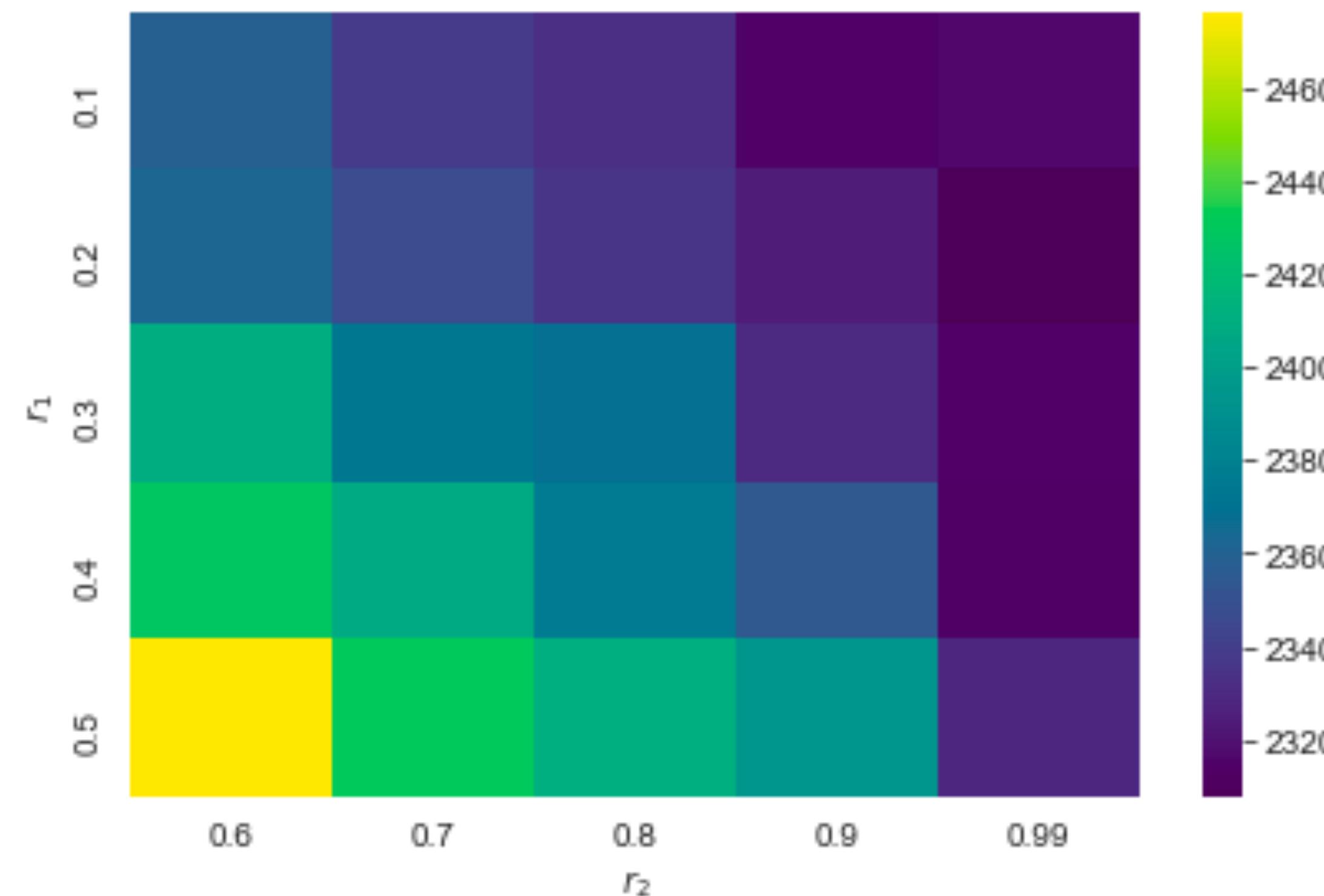
# Choose the learning rate $\rho$ dynamically

The best 10% of the samples  
have the 90% of the total weight



**The best 10% of the samples have the 90% of the total weight**

**The ratios are obtained experimentally**



# Linear ordering problem

Usually consider orderings

- Find the permutation of rows and columns that minimizes the sum of the lower triangle

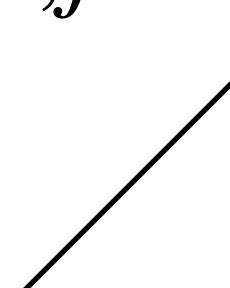
$$\min_{\sigma \in S_n} f(\sigma) = \sum_{i=1}^n \sum_{j=1}^{i-1} a_{\sigma^{-1}(i), \sigma^{-1}(j)}$$

# Permutation Flowshop Scheduling Problem, PFSP

- Minimize the *makespan*, the completion time of the last job on the last machine
- $\min_{\sigma \in S_n} f(\sigma) = C_{n,M}$ 
  - $C_{1,j} = 0 \quad j \in [M], \quad C_{i,1} = 0 \quad i \in [n]$
  - $C_{i,j} = p_{\sigma^{-1}(i),j} + \max\{C_{i-1,j}, C_{i,j-1}\} \quad i \in \{2, \dots, n\}, j \in \{2, \dots, M\}$

# Permutation Flowshop Scheduling Problem, PFSP

- Minimize the *makespan*, the completion time of the last job on the last machine
- $\min_{\sigma \in S_n} f(\sigma) = C_{n,M}$ 
  - $C_{1,j} = 0 \quad j \in [M], \quad C_{i,1} = 0 \quad i \in [n]$
  - $C_{i,j} = p_{\sigma^{-1}(i),j} + \max\{C_{i-1,j}, C_{i,j-1}\} \quad i \in \{2, \dots, n\}, j \in \{2, \dots, M\}$



# Permutation Flowshop Scheduling Problem, PFSP

- Minimize the *makespan*, the completion time of the last job on the last machine

- $\min_{\sigma \in S_n} f(\sigma) = C_{n,M}$ 
  - $C_{1,j} = 0 \quad j \in [M], \quad C_{i,1} = 0 \quad i \in [n]$
  - $C_{i,j} = p_{\sigma^{-1}(i),j} + \max\{C_{i-1,j}, C_{i,j-1}\} \quad i \in \{2, \dots, n\}, j \in \{2, \dots, M\}$



# Permutation Flowshop Scheduling Problem, PFSP

- Minimize the *makespan*, the completion time of the last job on the last machine

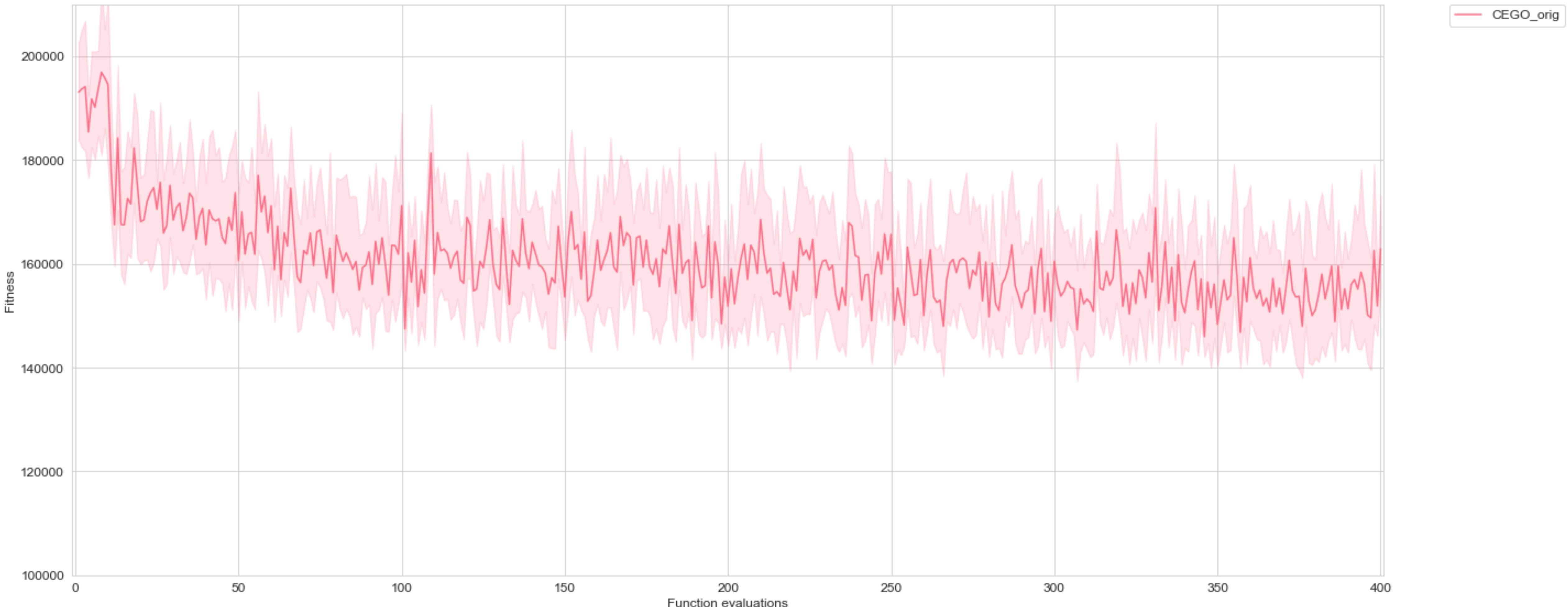
- $\min_{\sigma \in S_n} f(\sigma) = C_{n,M}$ 
  - $C_{1,j} = 0 \quad j \in [M], \quad C_{i,1} = 0 \quad i \in [n]$
  - $C_{i,j} = p_{\sigma^{-1}(i),j} + \max\{C_{i-1,j}, C_{i,j-1}\} \quad i \in \{2, \dots, n\}, j \in \{2, \dots, M\}$



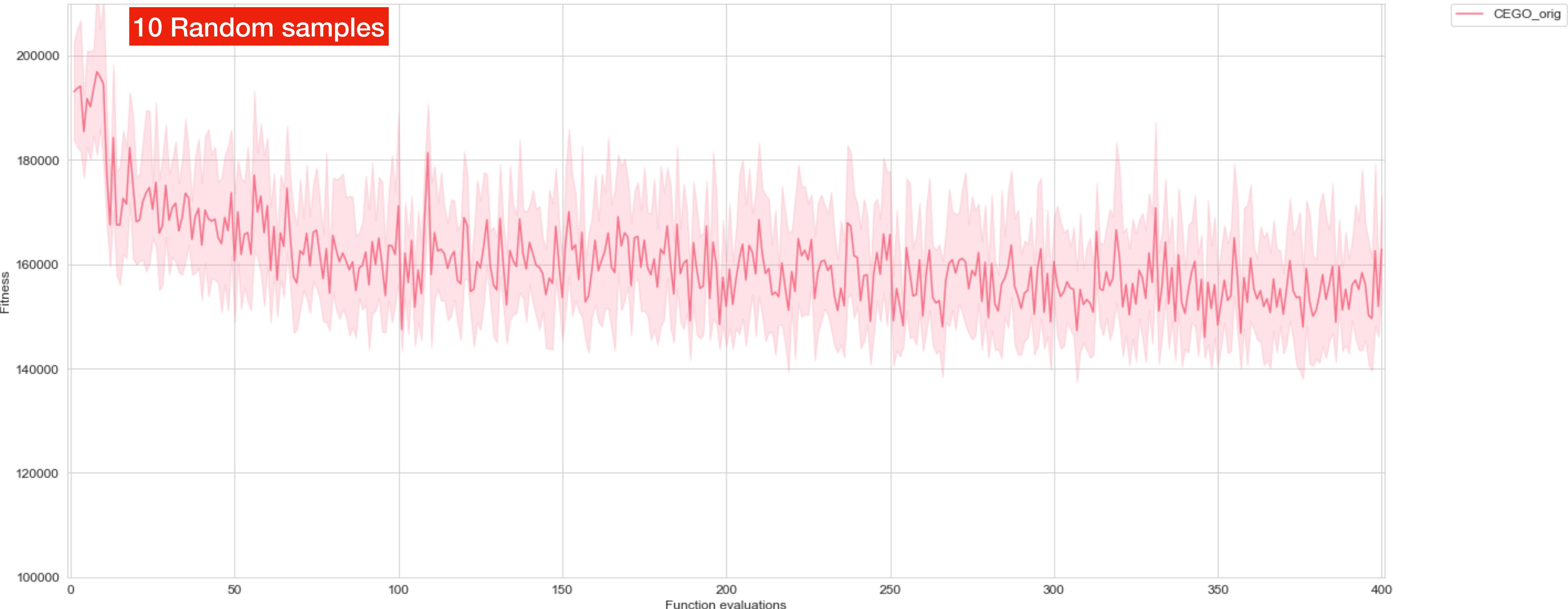
# State-of-the-art algorithm

- EGO/CEGO use Gaussian processes as a surrogate for the function
- CEGO is the state-of-the-art for expensive black-box permutation problems
  - Incremental sample
  - Maximize the E.I. criterion to predict the new point to evaluate
  - The surrogate model is explored with a GA

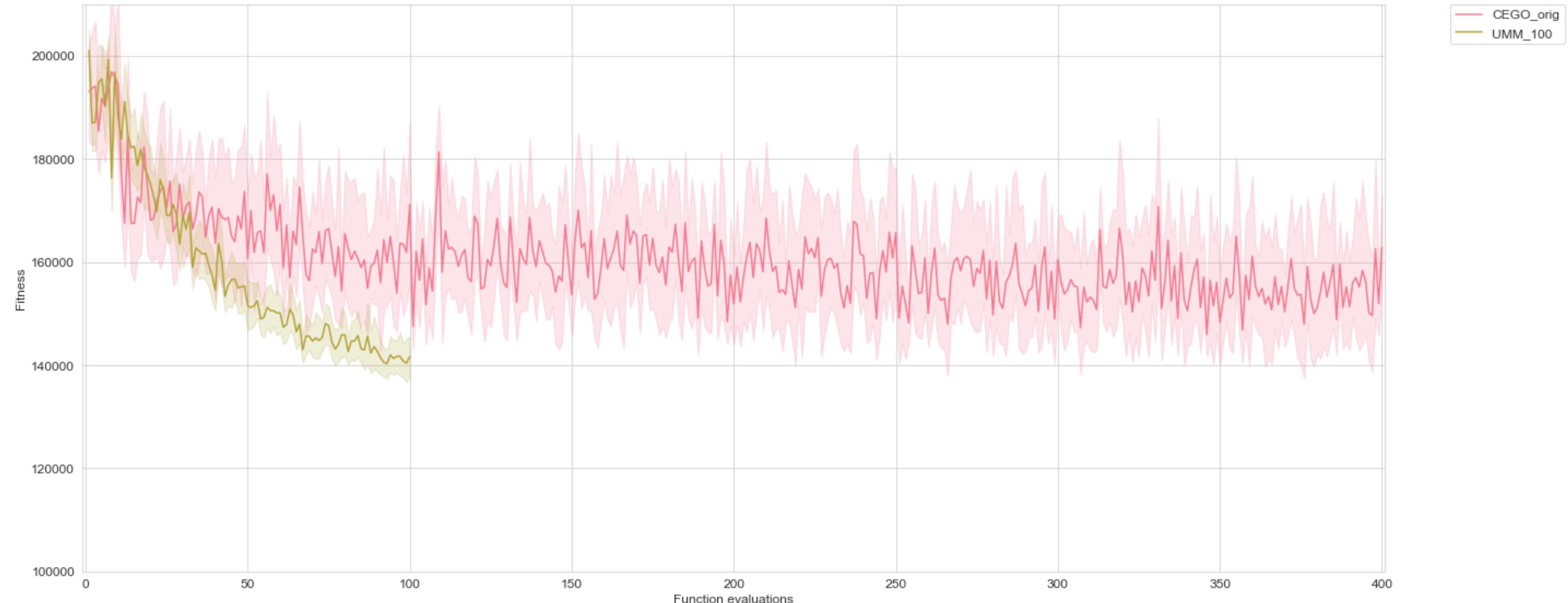
# CEGO, the state-of-the-art Minimization problem

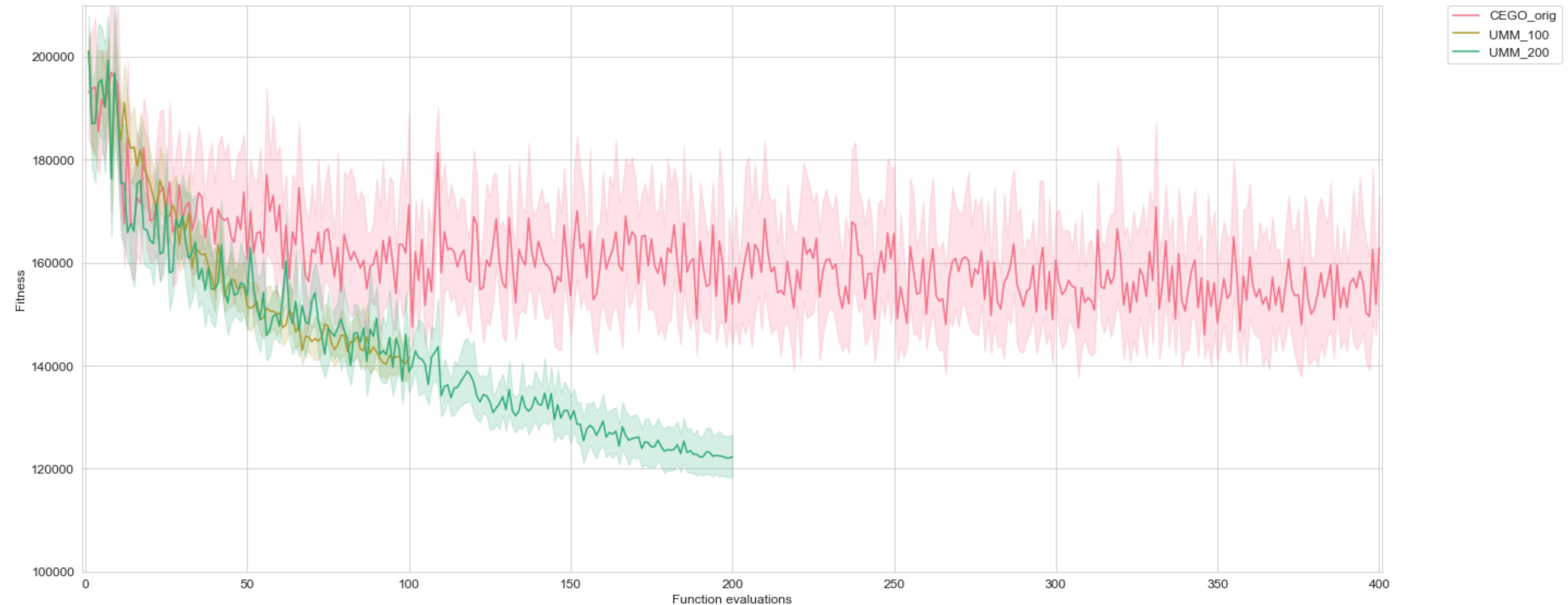


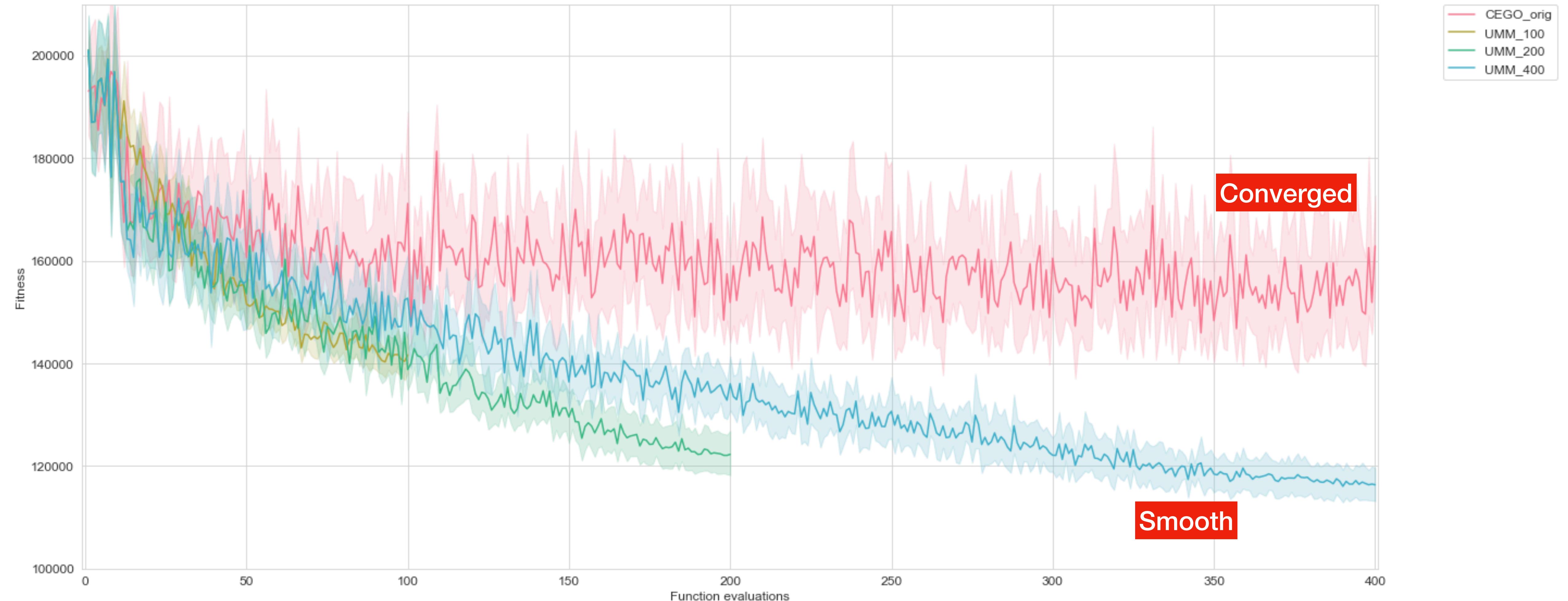
# CEGO, the state-of-the-art Minimization problem



# The variance of the model decreases linearly

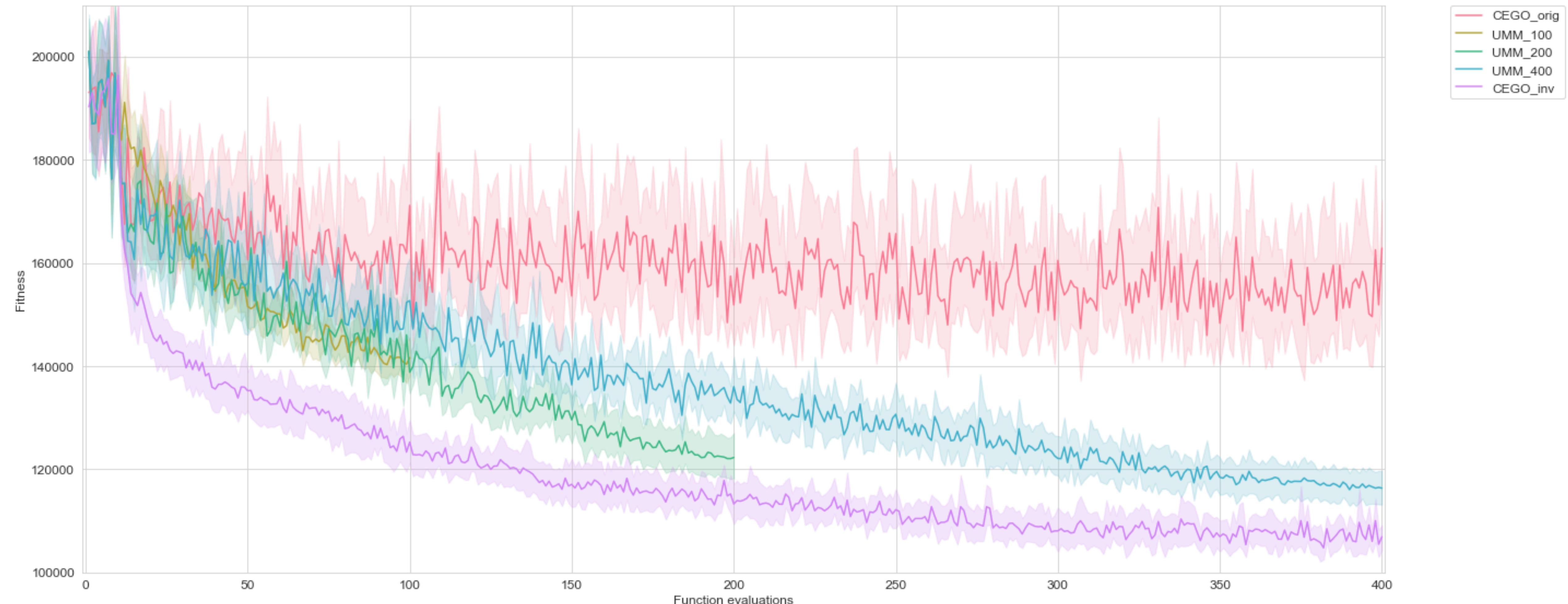






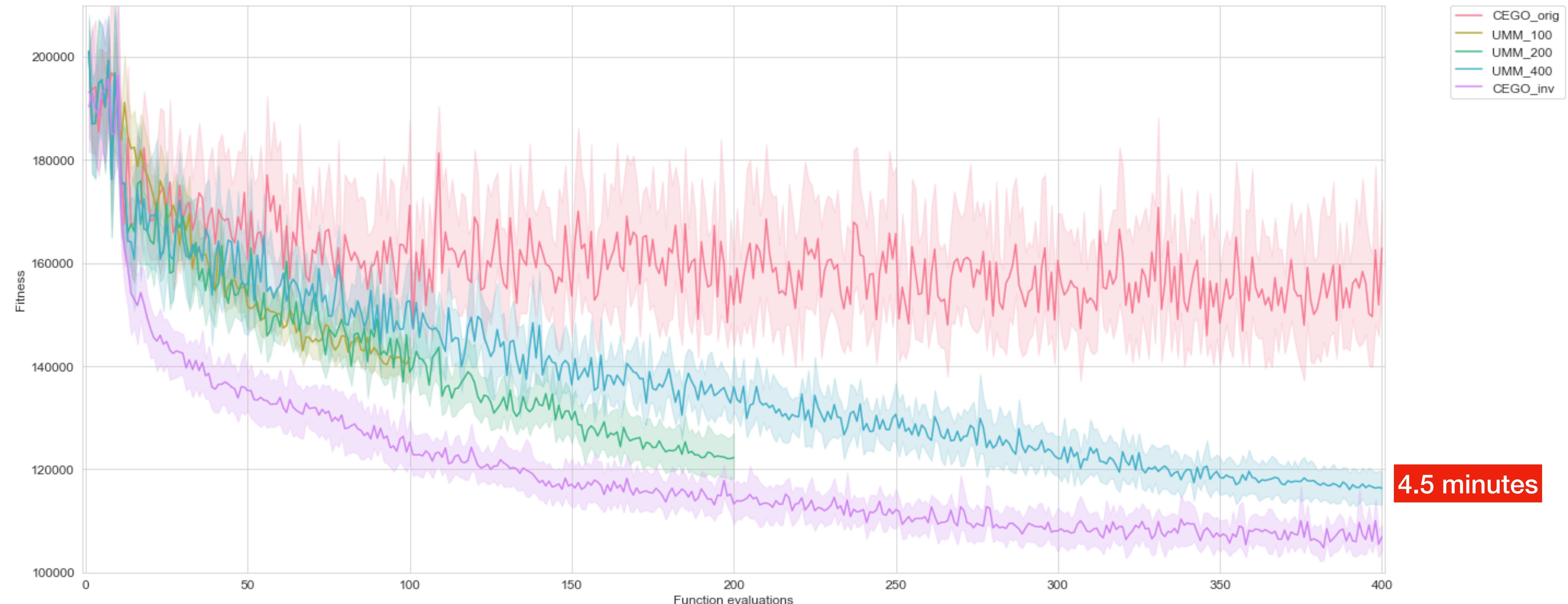
# Rankings and orderings

## Not just an implementation detail



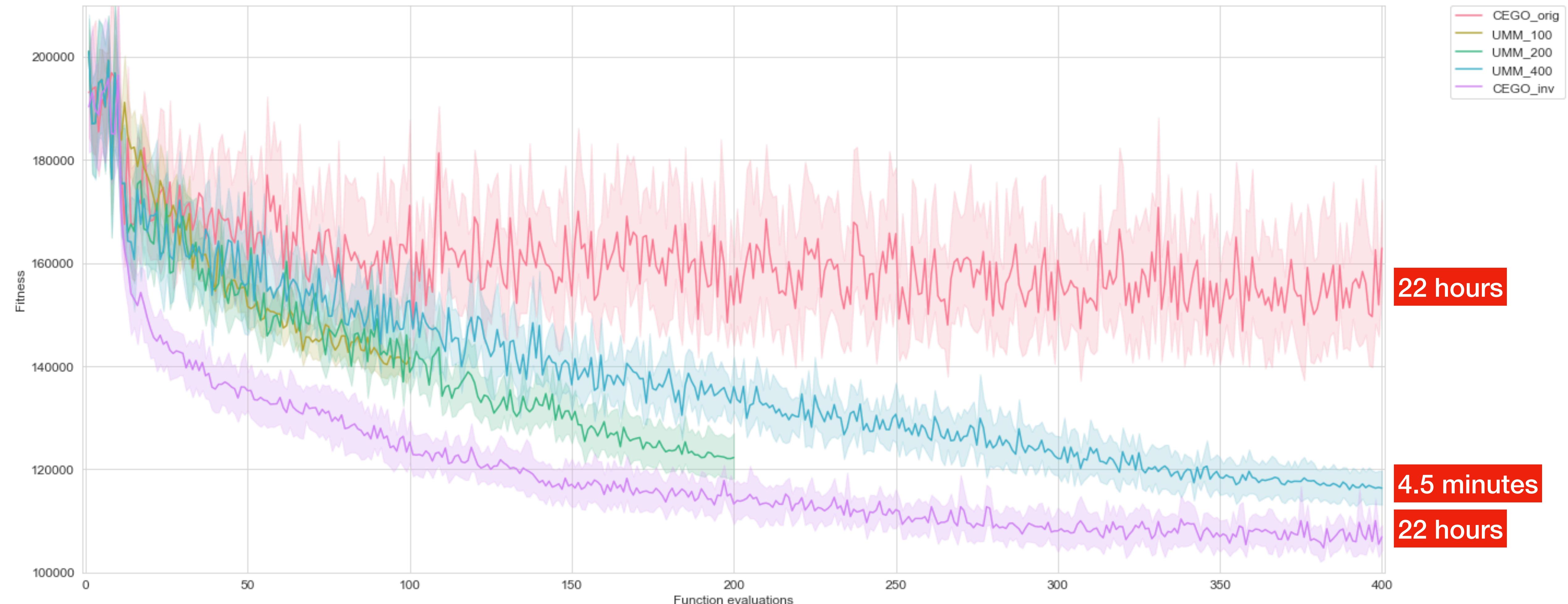
# Rankings and orderings

## Not just an implementation detail

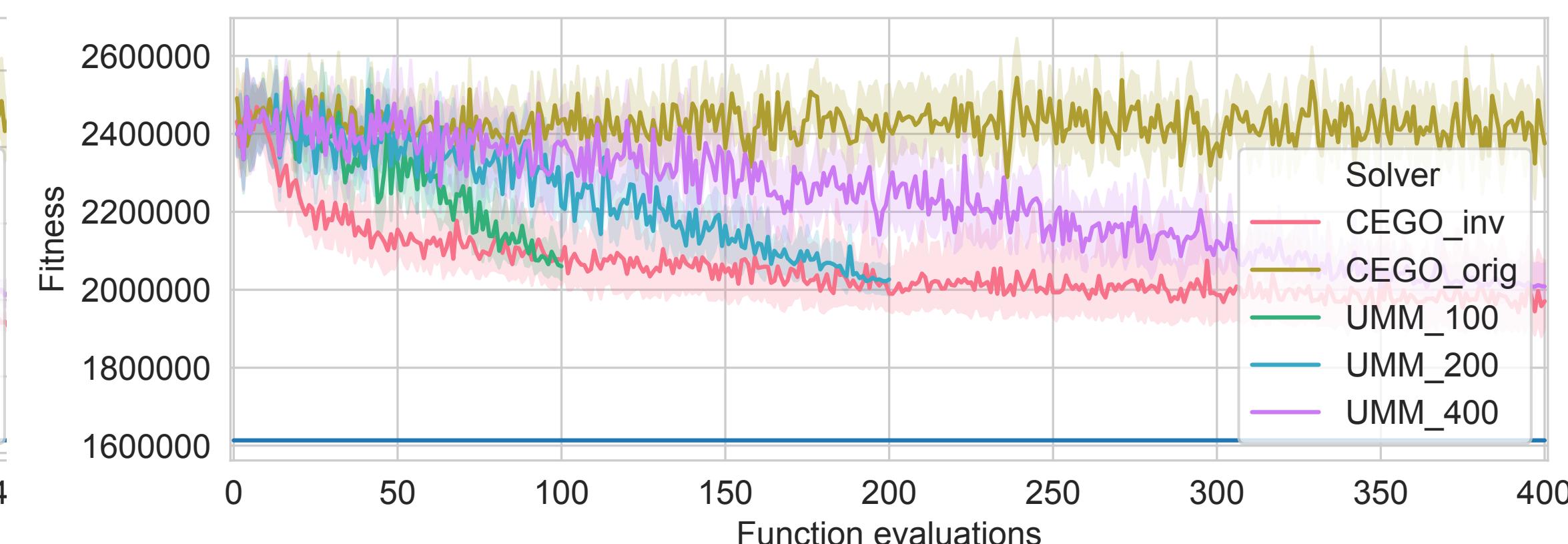
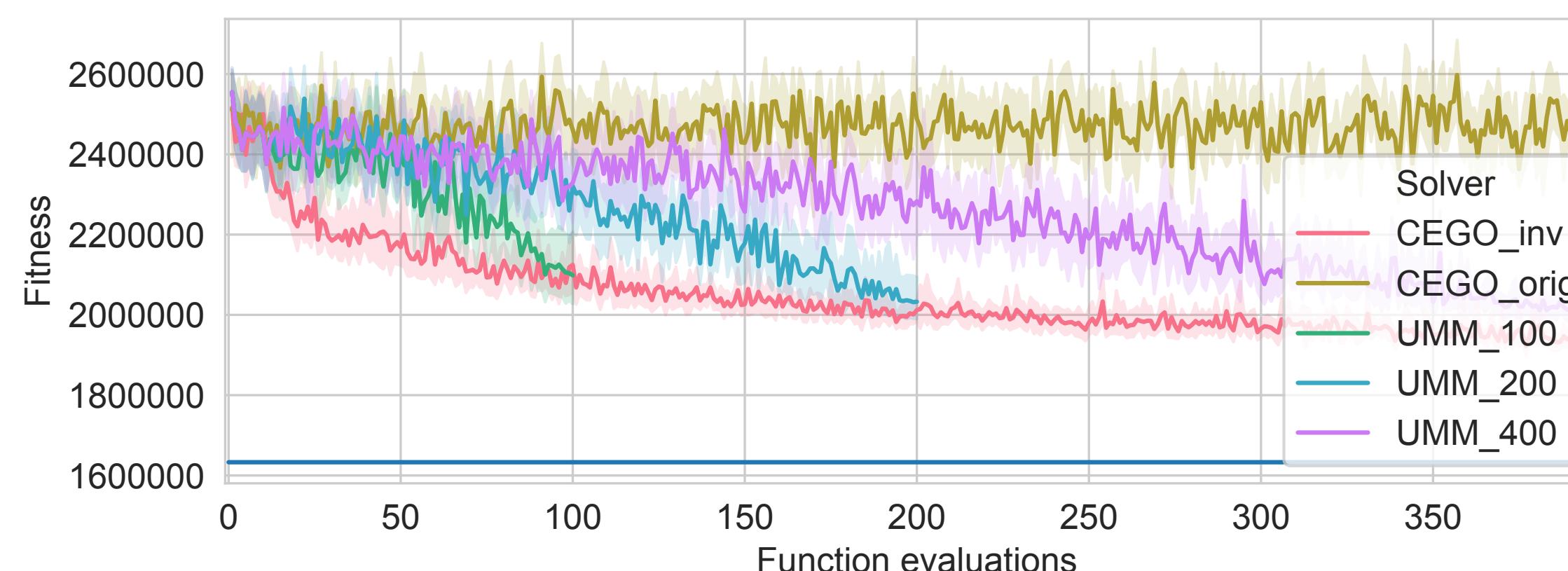
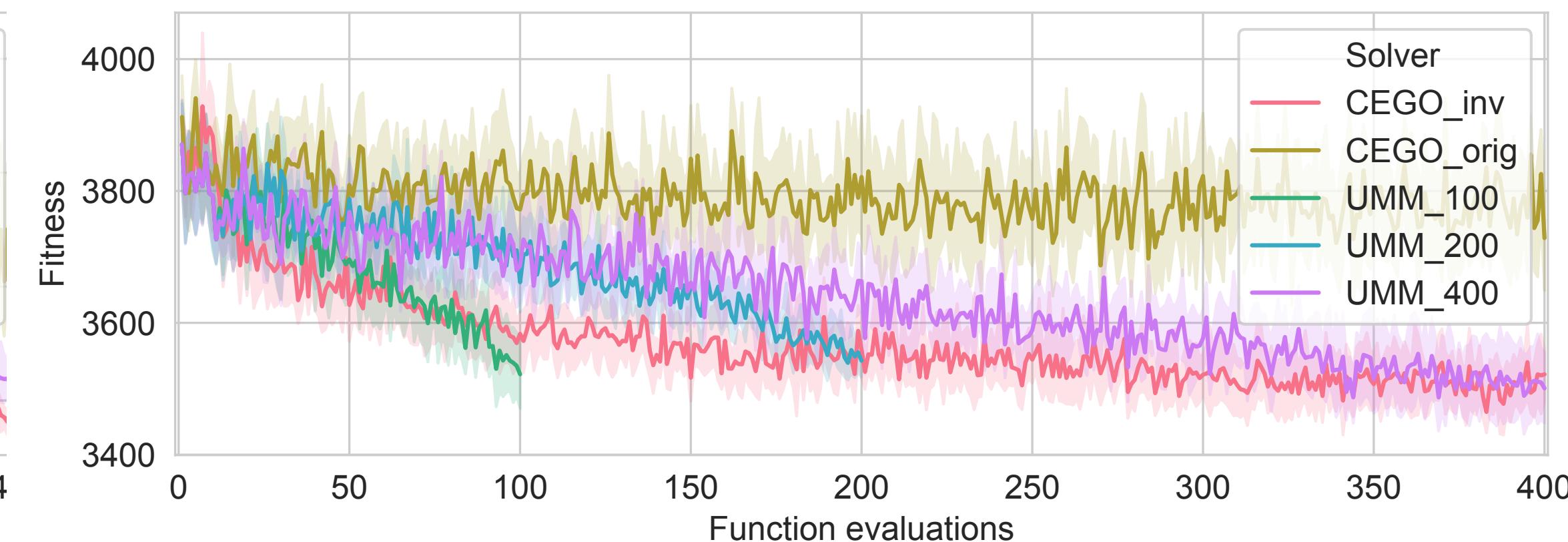
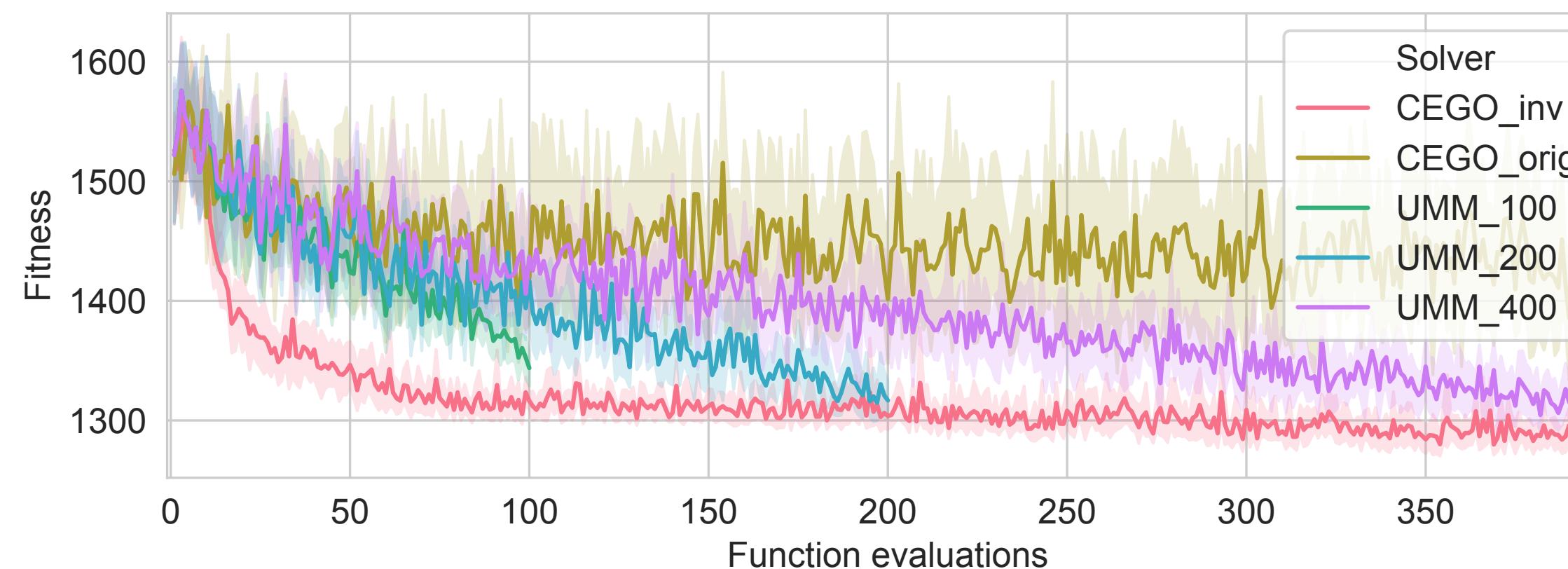


# Rankings and orderings

## Not just an implementation detail



# UMM outperforms the original CEGO It is competitive with the modified version



# References

- F. Collas and E. Irurozki. “Concentric Mallows mixtures for top-k rankings: sampling and identifiability”. In: International Conference on Machine Learning (ICML). 2021
- E. Irurozki, B. Calvo, and J. Lozano. “PerMallows: An R package for mallows and generalized mallows models”. In: Journal of Statistical Software 71 (2019)
- E. Irurozki, J. Lobo, A. Perez, and J. D. Ser. “Online Ranking with Concept Drifts in Streaming Data”. Submitted to European Conference on Machine Learning (ECML). 2021

# Conclusions

We present UMM

- A new framework for black-box expensive optimization
- Probabilistic, incremental sample
- Fast, efficient
  - Outperforms the state-of-the-art
  - Competitive with the modified version of the state state-of-the-art
- Permutations are not just ordered vectors
- Ad-hoc strategies for permutations often outperform the adaptations