UNIVERSITÄT
LEIPZIG

Hyperparameter Optimization Methods for the H-SPPBO Metaheuristic

Applied to the Dynamic Traveling Salesperson Problem

Leipzig, 30.03.23
Daniel Werner

Fakultät für Mathematik und Informatik

Schwarmintelligenz und Komplexe Systeme

CONTENT

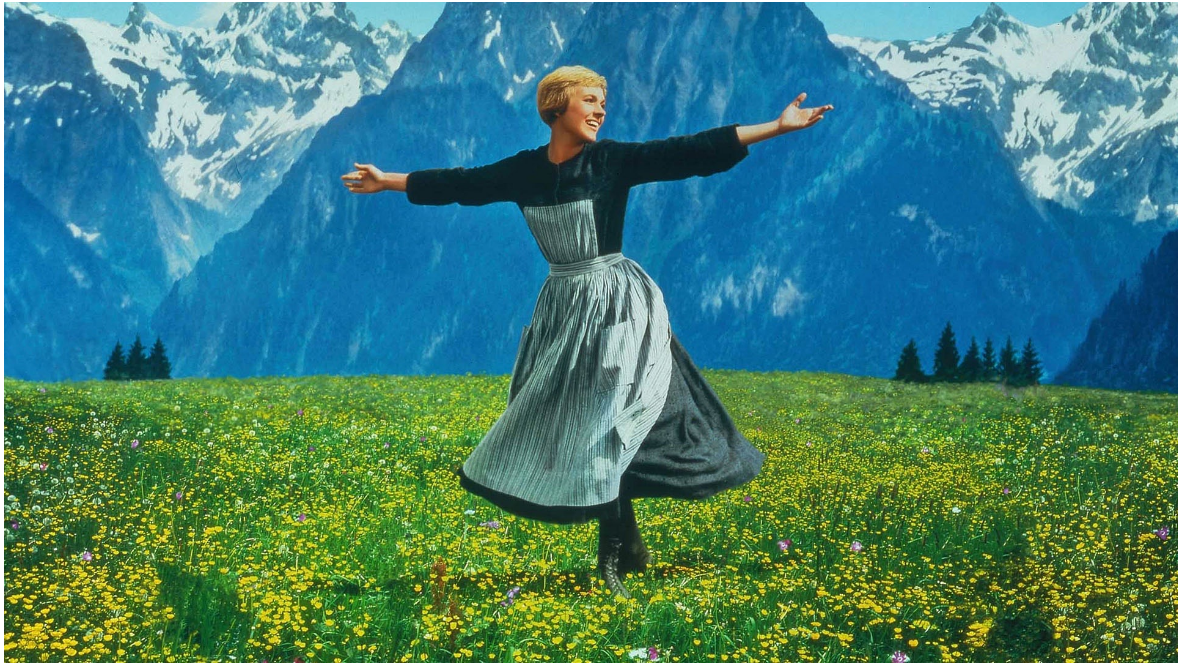
1. Motivation, Problem, Approach

WHY, WHAT AND HOW?

MOTIVATION

- **Combinatorial problems** everywhere:
  - Transporting, logistics, digital traffic routing
  - Complex problems (NP-hard) need metaheuristics
- No free lunch theorem → No single metaheuristic is the best for all problems
- **Metaheuristic Frameworks**: Streamlining algorithm design
  - Simple Probabilistic Population-Based Optimization (SPPBO)
  - Adaptable to many problems through parameters

MOTIVATION



?

PROBLEM

- **Problem 1**: Feature- and parameter-rich metaheuristic frameworks
  - Not feasibly tuned by hand
  - Only **optimal parameters** yield **optimal solutions**
- **Problem 2**:

  Real world rarely static, **dynamic problems needed**
  - TSP → DTSP

APPROACH

- **Solution 1**: Hyperparameter Optimization (HPO)
  - Methods from **Machine Learning**
  - Applied to Parameter Tuning of Metaheuristics
- **Solution 2**: Hierarchical Simple Probabilistic Population-Based Optimization
  - Designed using SPPBO and H-PSO
  - Solves the DTSP and **detects change**

RESEARCH QUESTIONS

What is the ideal HPO method for the H-SPPBO algorithm?

↓

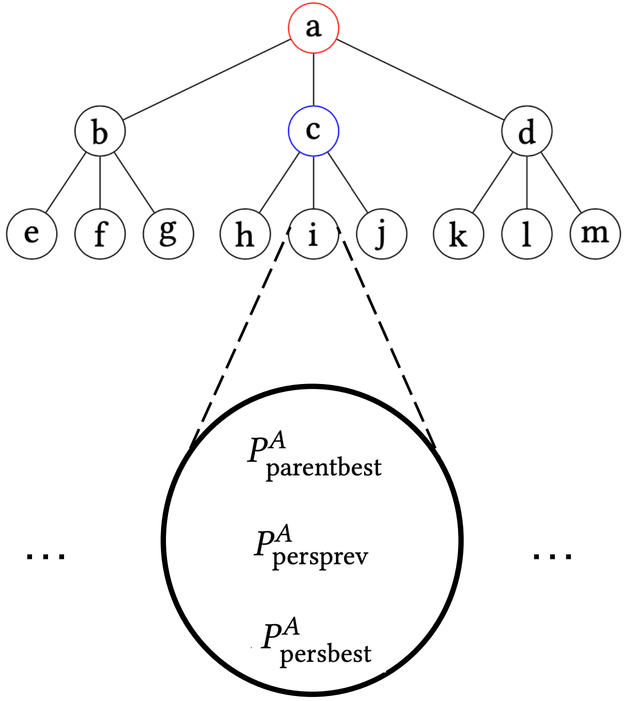
Which sets of parameters yield the best results for a given DTSP instance?

2. Theoretical Background

WHAT DO I NEED TO KNOW?

HIERARCHICAL SIMPLE PROBABILISTIC POPULATION-BASED OPTIMIZATION (H-SPPBO)

- SCEs: Solution creating entities (≈ particles, ants)
  - Organized in a **hierarchical** m-ary tree
  - Defines **neighborhood relation** (range)
- **Three populations** influence solutions
  - Personal **previous** solution P^A_{persprev}
  - Personal best solution P^A_{persbest}
  - Parent best solution $P^A_{\text{parentbest}}$



HIERARCHICAL SIMPLE PROBABILISTIC POPULATION-BASED OPTIMIZATION (H-SPPBO)

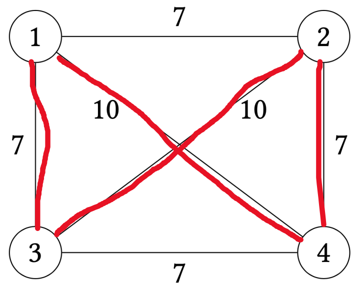
- Solutions created using **probabilistic term** τ
  - Start at random city node i
  - Set of unvisited cities U, iterate for: $k \in U$

$$\tau_{ik}(A) = \underbrace{\Big[w_{\text{rand}} + w_{\text{persprev}} \cdot s_{ik}(P_{\text{persprev}}) + w_{\text{persbest}} \cdot s_{ik}(P_{\text{persbest}}) + w_{\text{parentbest}} \cdot s_{ik}(P_{\text{parentbest}})\Big]^{\alpha}}_{\text{Stochastic}} \cdot \underbrace{\eta_{ik}^{\beta}}_{\text{Heuristic}}$$

HIERARCHICAL SIMPLE PROBABILISTIC POPULATION-BASED OPTIMIZATION (H-SPPBO)

- Solutions created using **probabilistic term** τ
  - Start at random city node i
  - Set of unvisited cities U, iterate for: $k \in U$

$$\tau_{ik}(A) = \underbrace{\left[w_{\text{rand}} + w_{\text{persprev}} \cdot s_{ik}(P_{\text{persprev}}) + w_{\text{persbest}} \cdot s_{ik}(P_{\text{persbest}}) + w_{\text{parentbest}} \cdot s_{ik}(P_{\text{parentbest}})\right]^{\alpha}}_{\text{Stochastic}} \cdot \underbrace{\eta_{ik}^{\beta}}_{\text{Heuristic}}$$



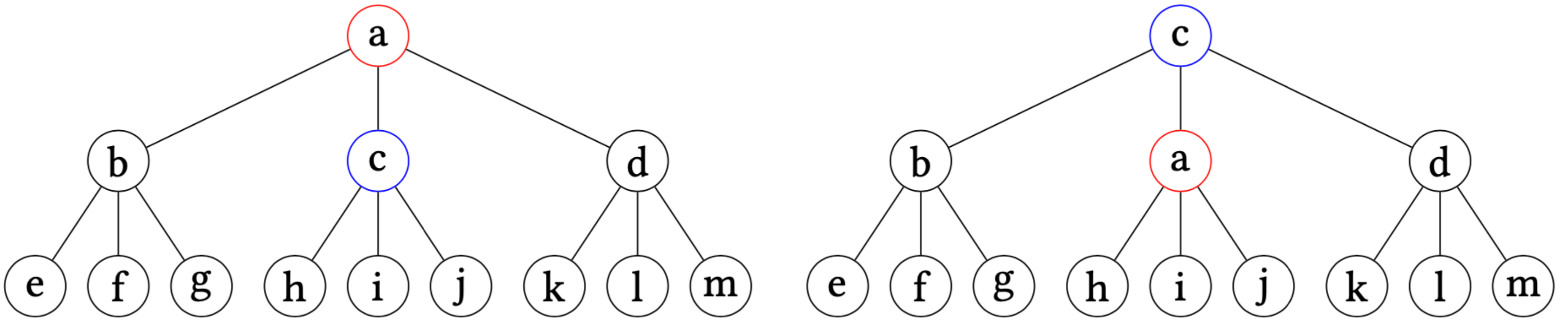
P_{persprev} (1,4,2,3)

i k
(4,2)
$+w_{\text{persprev}}$

i k
~~(4,3)~~

HIERARCHICAL SIMPLE PROBABILISTIC POPULATION-BASED OPTIMIZATION (H-SPPBO)

- SCE tree updated with new solutions: **Top-Down, Breadth-First**



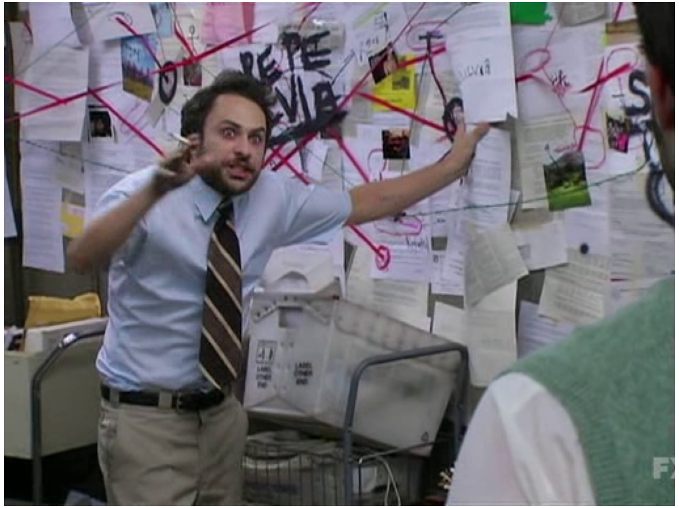
HIERARCHICAL SIMPLE PROBABILISTIC POPULATION-BASED OPTIMIZATION (H-SPPBO)

- **Detect** changes using **tree swaps**
  - Percentage of SCEs swapped per iteration $\theta \in [0, 1]$

- Threshold triggers **change handling procedure**: two strategies
  1. **Full** reset of all personal best solutions
  2. **Partial** reset of personal best solutions (from third tree level down)

PARAMETER OPTIMIZATION FOR METAHEURISTICS

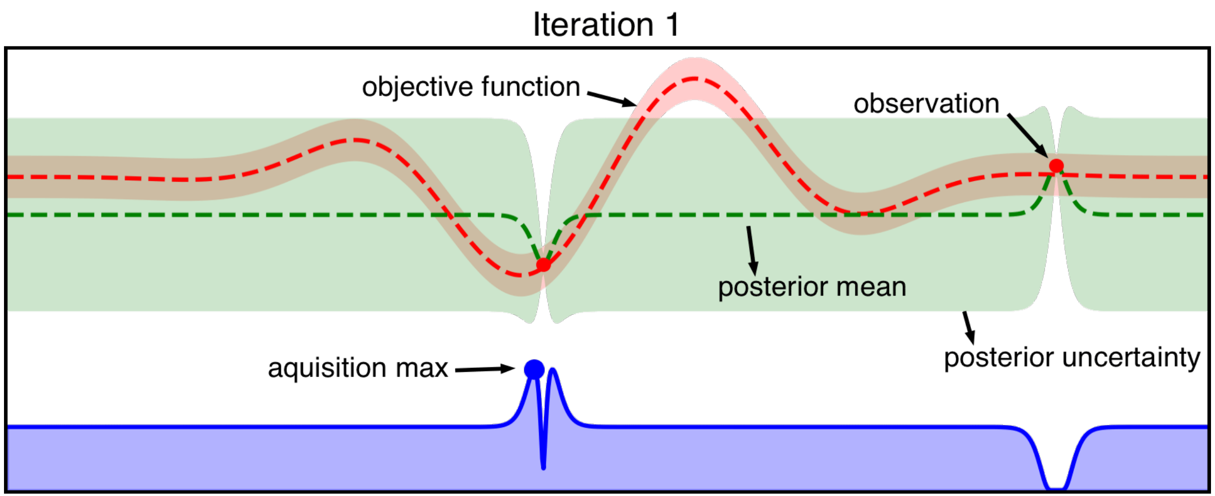
- H-SPPBO has many parameters:
  - $w_{\text{persprev}}, w_{\text{persbest}}, w_{\text{parentbest}} \geq 0$
  - $\alpha, \beta \geq 0$
  - $\theta \in [0, 1]$
  - change handling procedure (full/partial)
- Performance relies on **good parameters**
  - Solution: Manual Parameter Tuning
  - Better Solution: HPO

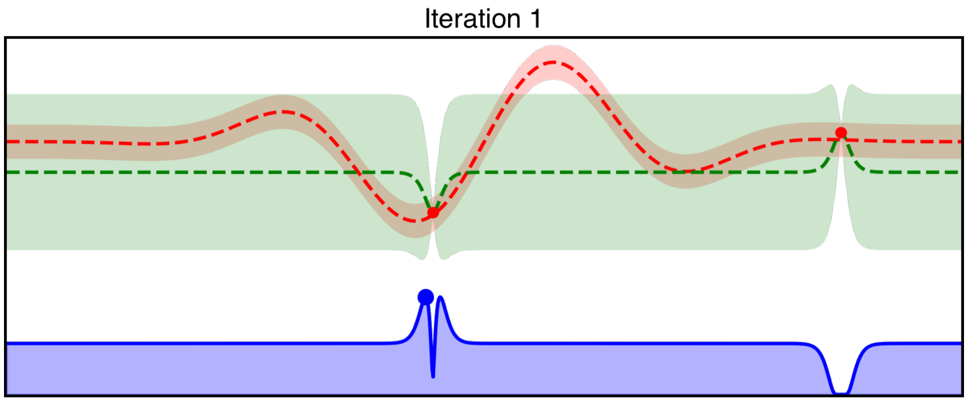


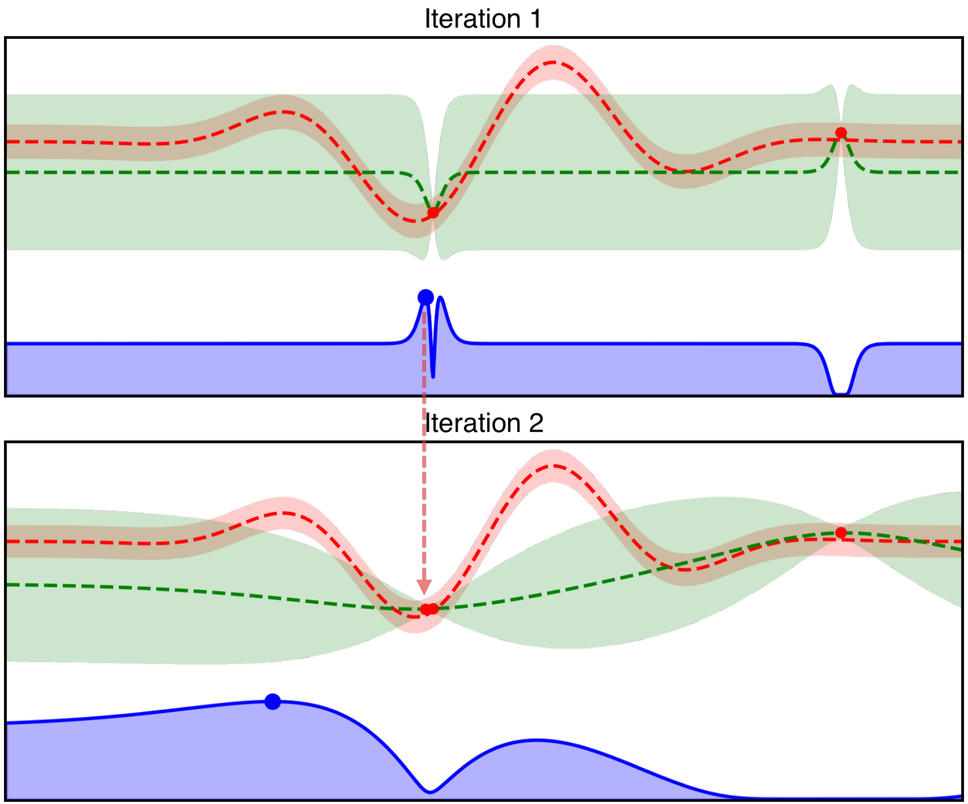
HPO: METHODS

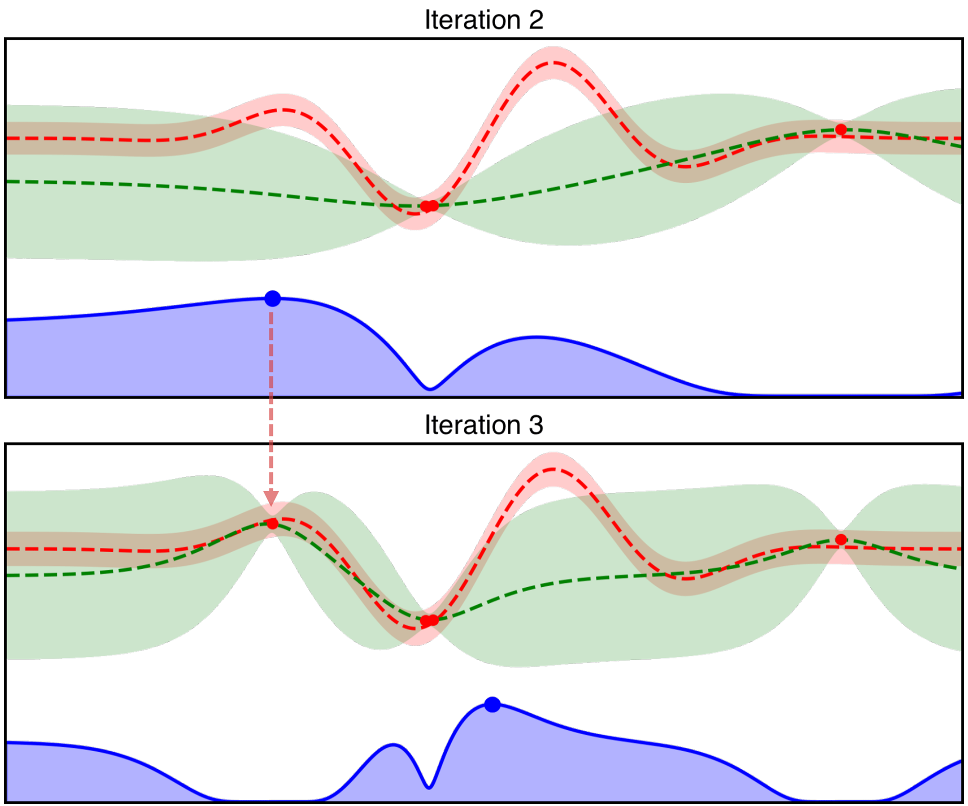
- Model-Free: Random Search (RS)
  - Randomized Brute Force approach (uniform samples)
- Model-Based: Bayesian Optimization (BO)
  - Algorithm framework for global optimization
  - **Bayes' theorem**: Incorporates **prior knowledge** into search
  - Two key components: **Surrogate Model** and **Acquisition Function**

BAYESIAN OPTIMIZATION (BO): ALGORITHM




Iteration 1


Iteration 1
Iteration 2


Iteration 2
Iteration 3

3. Implementation

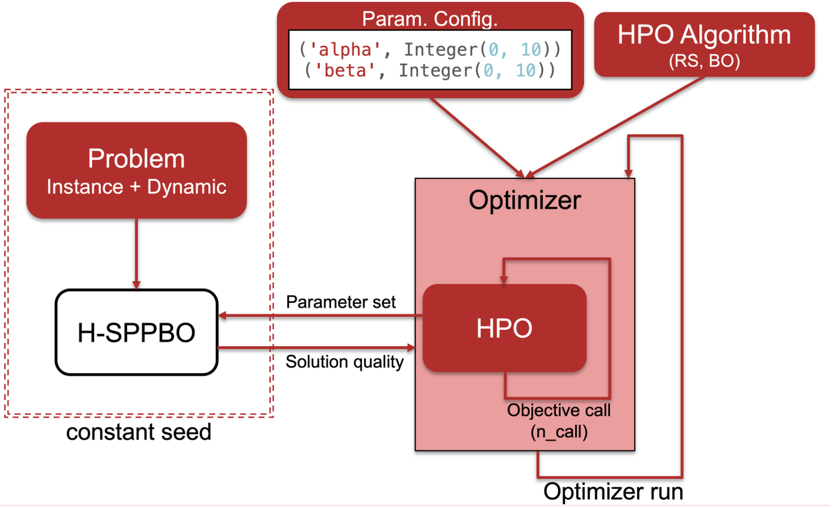
HOW IS THIS REALIZED?

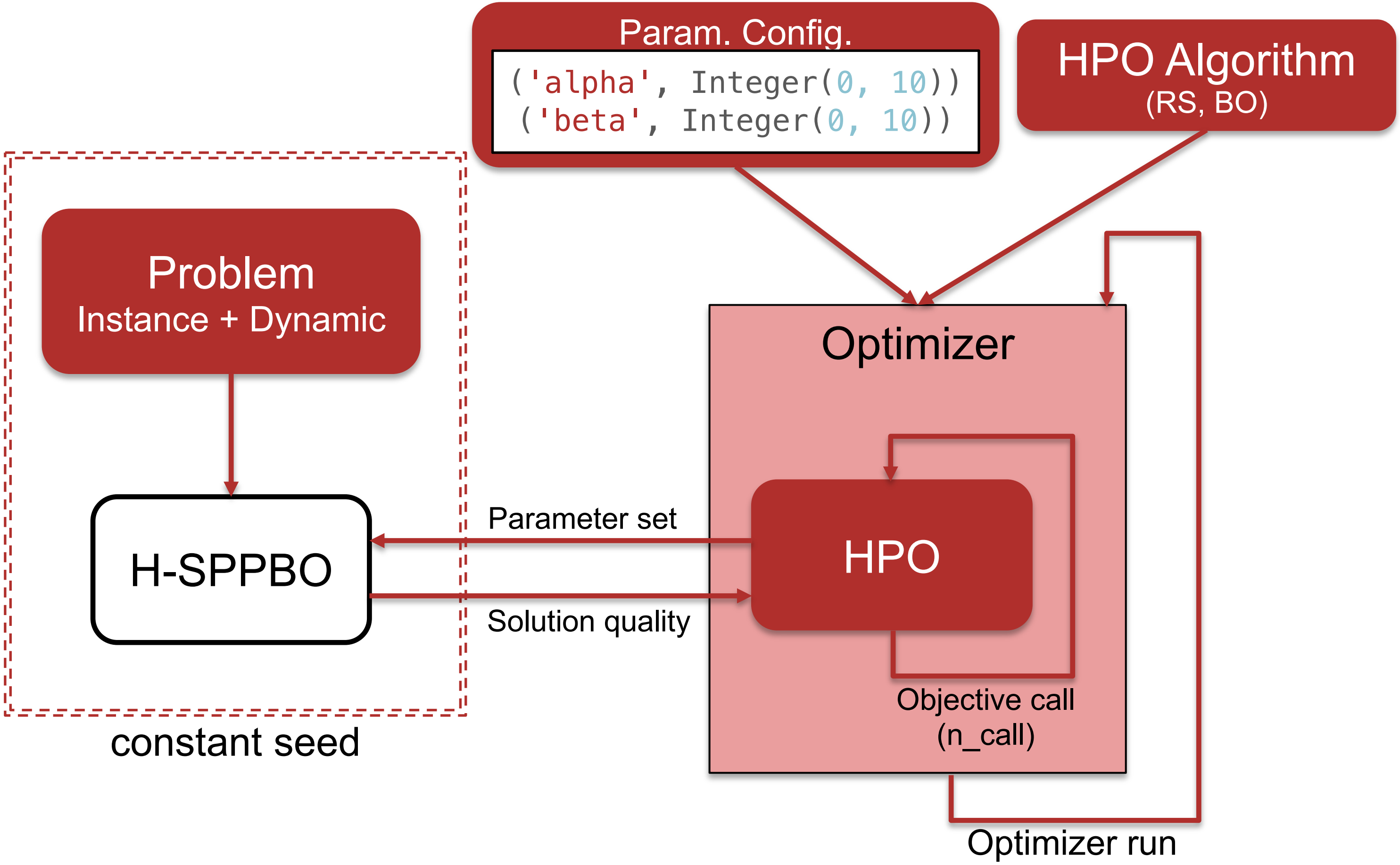
IMPLEMENTATION

- E**X**perimentation **F**ramework and (Hyper-)Parameter **Opt**imization for **Meta**heuristics (XF-OPT/META)
  - Python 3.10 or newer
  - Modular, expandable and object-oriented
- Three major, custom modules
  - Problem Interface + TSP Module
  - H-SPPBO Module
  - Optimizer Module

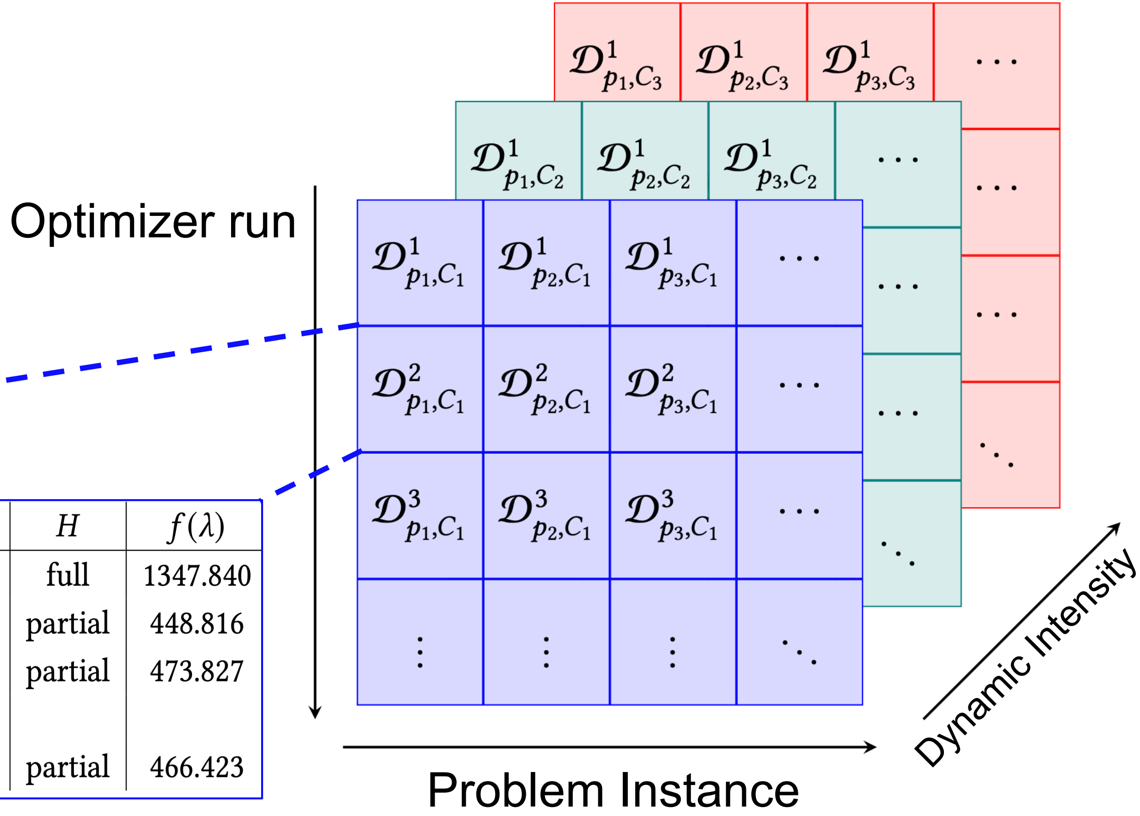
OPTIMIZER MODULE

- HPO based on *scikit-optimize (skopt)* library
  - Relies on popular *scikit-learn* package
- Deterministically **repeatable** optimizer runs
  - RNG seed fixed for all modules

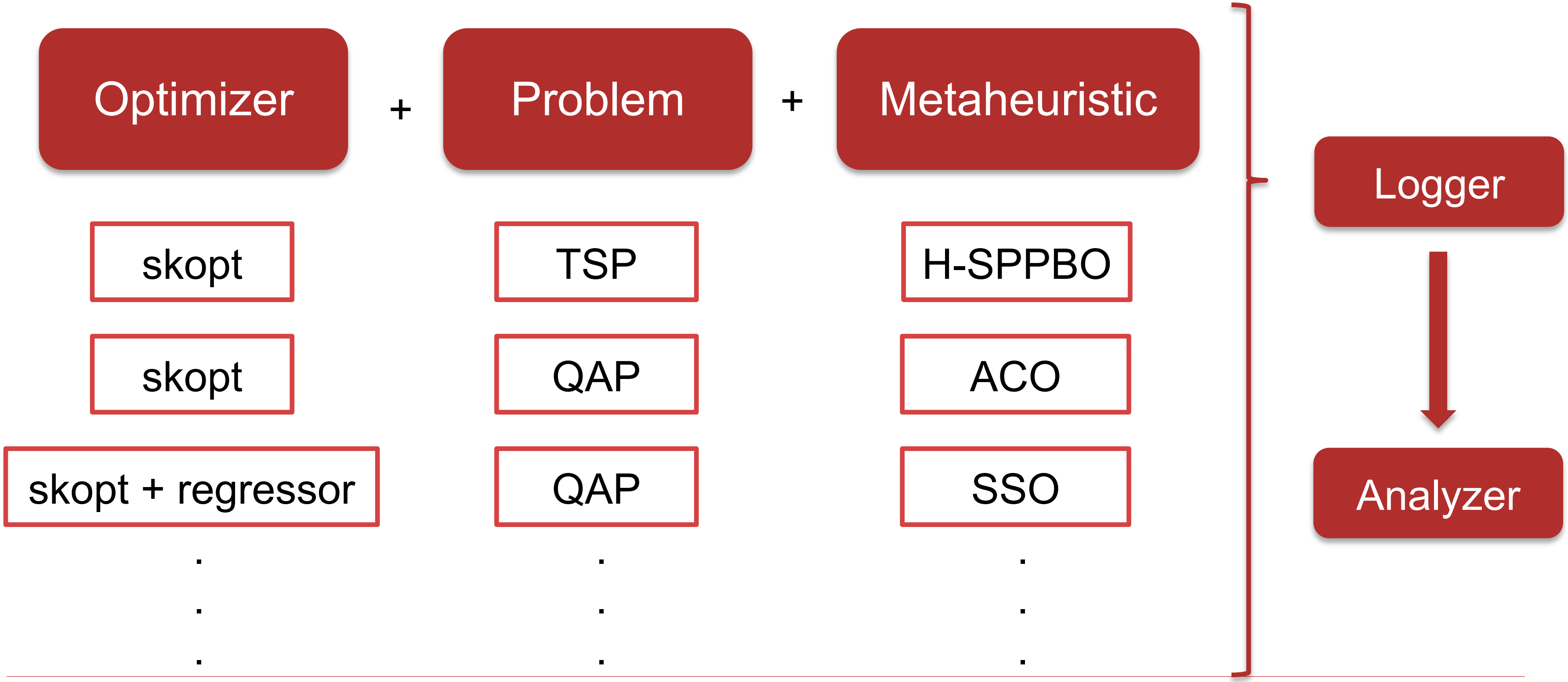



Param. Config.
('alpha', Integer(0, 10))
('beta', Integer(0, 10))
HPO Algorithm
(RS, BO)
Problem
Instance + Dynamic
H-SPPBO
Parameter set
Solution quality
Optimizer
HPO
Objective call
(n_call)
constant seed
Optimizer run

n_call	α	β	w_{persbest}	w_{persprev}	$w_{\text{parentbest}}$	θ	H	$f(\lambda)$
1	0	0	0.001	0.001	0.001	0.1	full	1347.840
2	10	10	0.763	0.930	0.989	0.416	partial	448.816
3	8	8	0.763	0.334	0.989	0.416	partial	473.827
...								
30	0	10	0.001	0.639	0.001	0.5	partial	466.423



FRAMEWORK VIEW AND WORKFLOWS

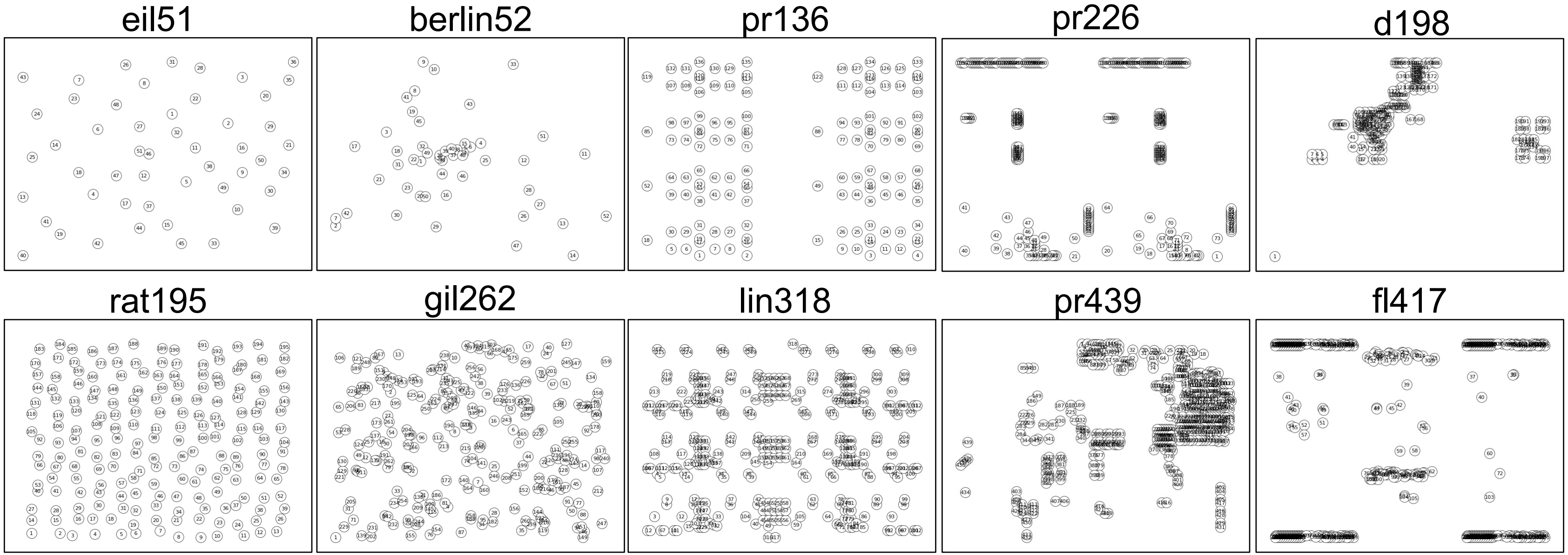


4. Experimental Design

WHAT DATA DO I NEED AND HOW DO I GET IT?

CHOICE OF PROBLEM INSTANCES

- Goal: **meaningful, disjoint subset** of problem instances from TSPLIB
- Two decision metrics: **Dimension** & **Placement characteristics**
- Dimension: Time as decision factor → smaller than 450 nodes
  - **Smaller** instances [50, 250]
  - **Larger** instances [250, 450]



smaller	eil51	berlin52	pr136	pr226	d198
larger	rat195	gil262	lin318	pr439	fl417
Group	1.	2.	3.	4.	5.
Structure	random, nearly regular	slightly clustered, otherwise random	artificial pattern, many medium clusters	few highly clustered	dispersed, highly clustered

CHOICE OF OPTIMIZATION METHODS

– Random Search & Bayesian Optimization

→ Four distinct HPO methods:

Surrogate Models

1. Random Search (**RS**)
2. Bayesian Optimization using Gaussian Process (**BO-GP**)
3. Bayesian Optimization using Extra Tress (**BO-ET**)
4. Bayesian Optimization using Gradient Boosted Regression Trees (**BO-GBRT**)

TESTING PROCEDURE

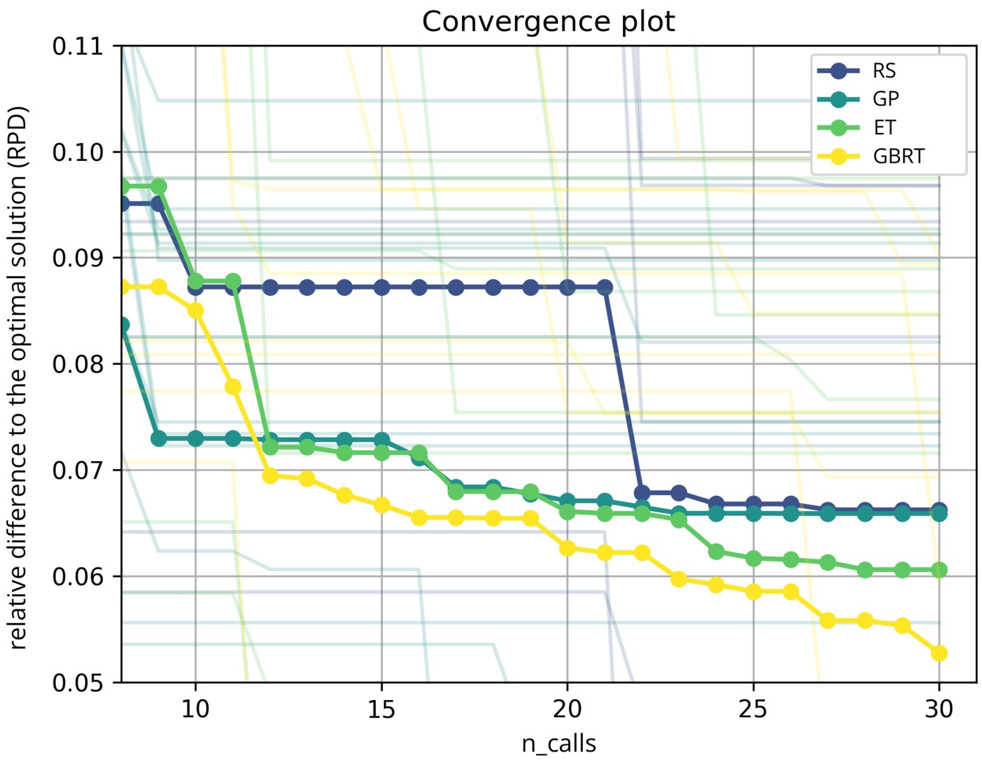
- Three test parts
  1. Find most appropriate HPO method for H-SPPBO
  2. Find optimal parameter sets for each problem instance
  3. Evaluate and compare HPO parameters with standard choices

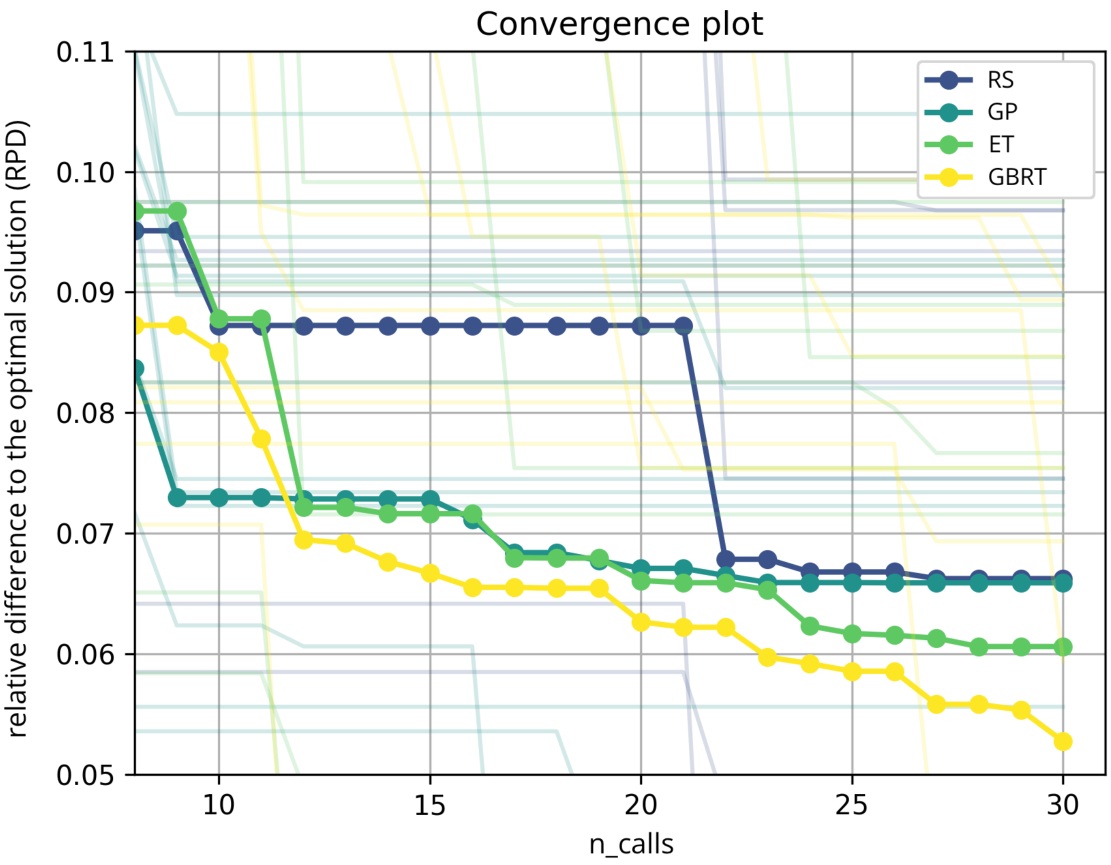
- Two servers (16/36 threads), taking ~3 months

5. Results

WHAT DOES THE DATA SAY?

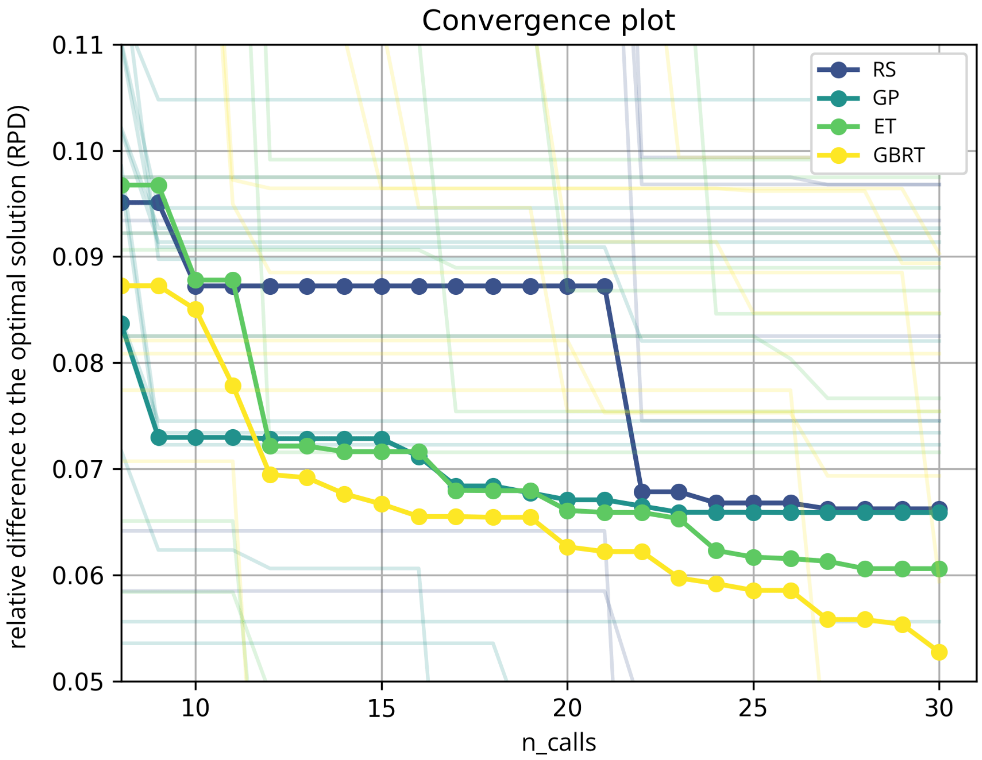
RESULTS: PART 1 - CONVERGENCE BEHAVIOR




Convergence plot
0.11
0.10
0.09
0.08
0.07
0.06
0.05
relative difference to the optimal solution (RPD)
10
15
20
25
30
n_calls
RS
GP
ET
GBRT

RESULTS: PART 1
CONVERGENCE BEHAVIOR

	RS	GP	ET	GBRT
AUC	1.484	1.298	1.268	1.190
min	0.058	0.060	0.051	0.045



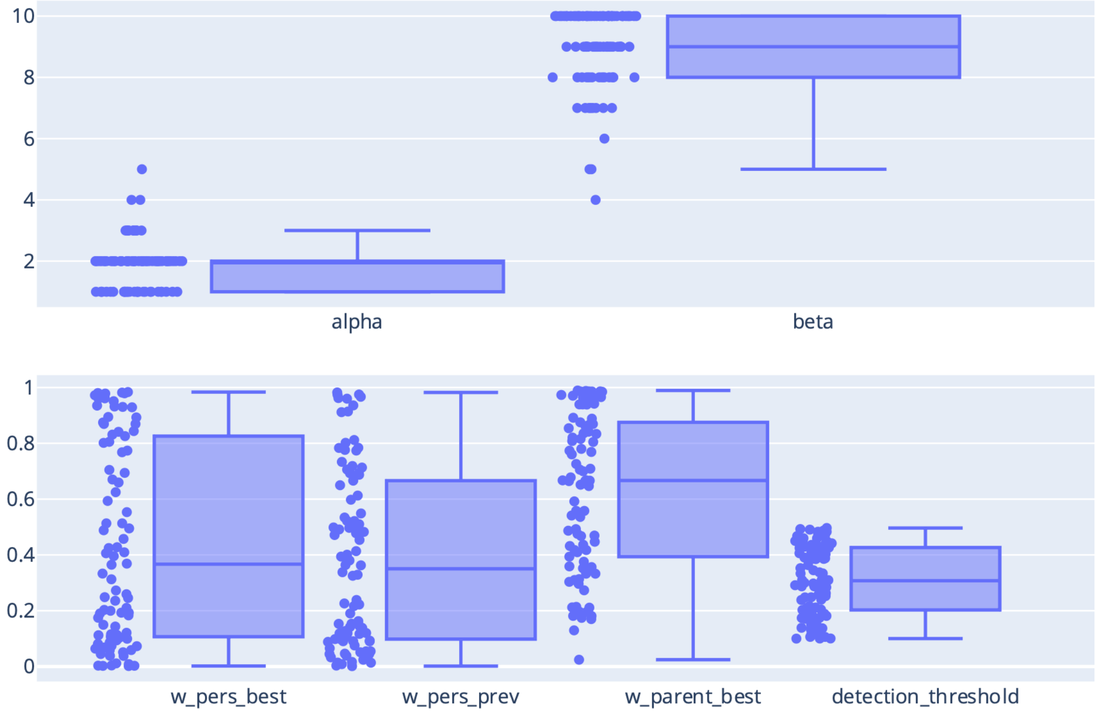
RESULTS: PART 1 - CONCLUSION

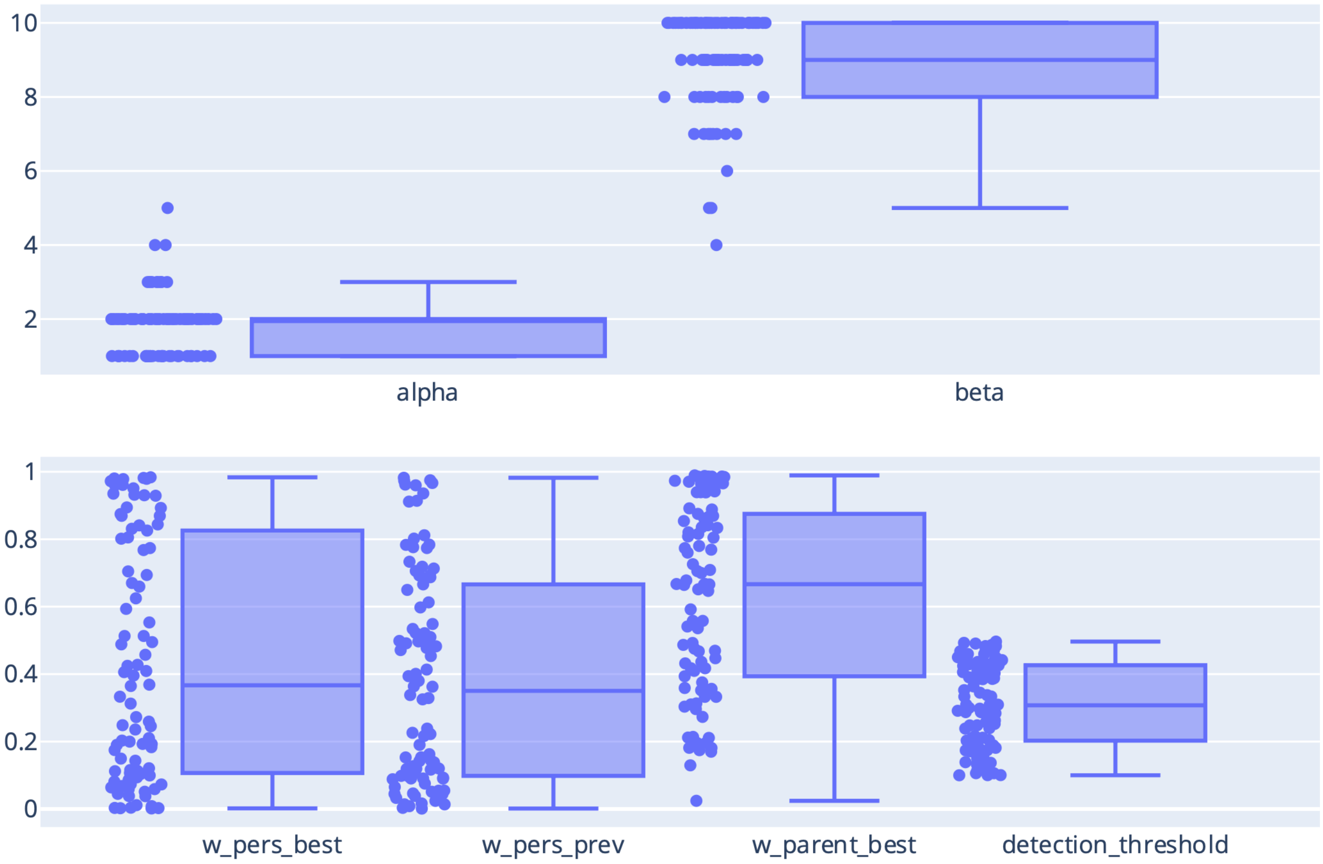
- GBRT performs best
  - Fastest convergence
  - Best solutions
  - Bonus: Parameter Importance through model
- GP: Good performance on larger instances
  - Narrow performance variance → reliable

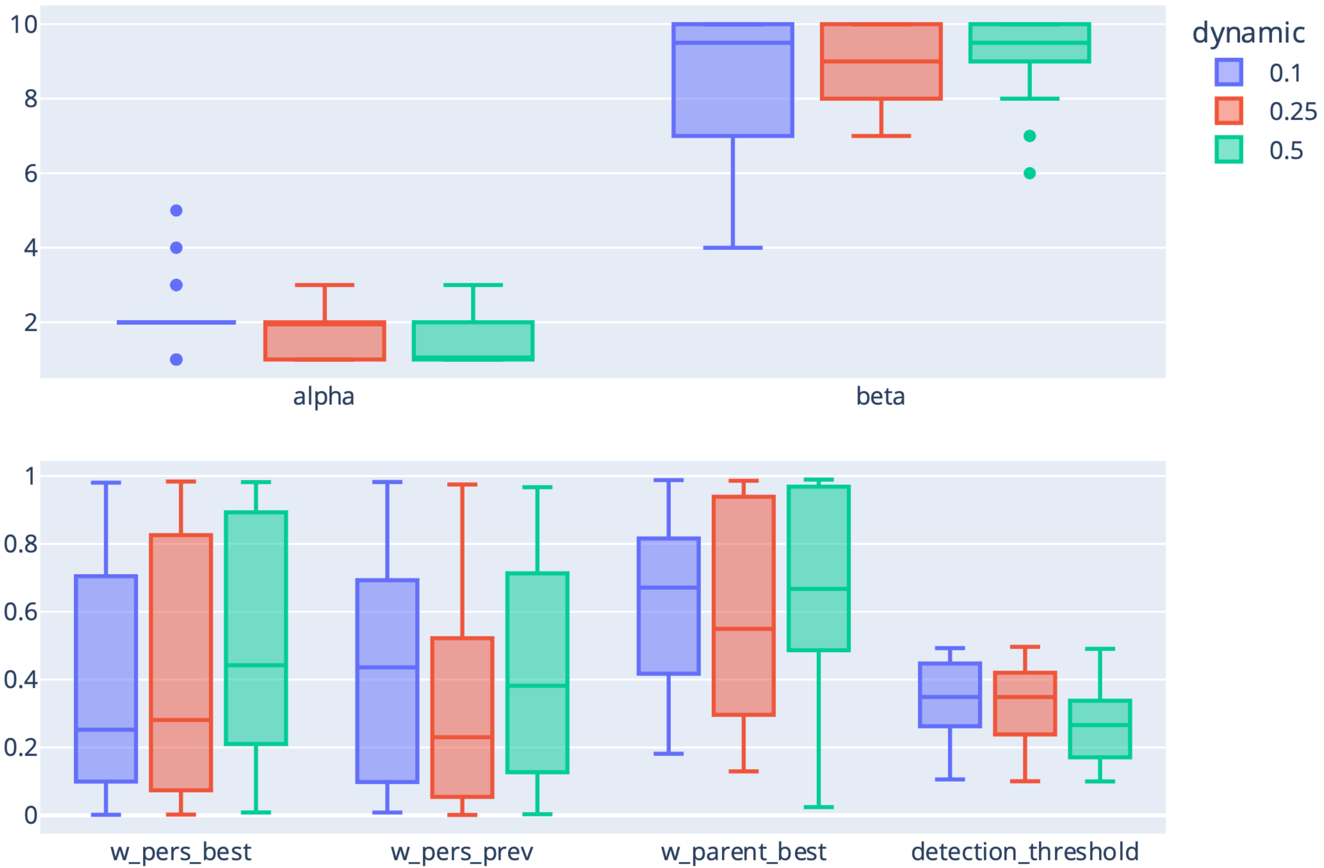
- GBRT chosen: convergence (speed) → shorter runtime

RESULTS: PART 2 – ROBUSTNESS

- Best parameter sets
  - 5 instances
  - 3 dynamics
  - 6 runs for each instance + dynamic
  - → 90 data points




10
8
6
4
2
alpha
beta
1
0.8
0.6
0.4
0.2
0
w_pers_best
w_pers_prev
w_parent_best
detection_threshold


10
8
6
4
2
alpha
beta
dynamic
0.1
0.25
0.5
1
0.8
0.6
0.4
0.2
0
w_pers_best
w_pers_prev
w_parent_best
detection_threshold

6. Conclusion

WHAT HAVE WE LEARNED?

CONCLUSION

- **HPO suitable** for H-SPPBO
  - → Good performance without deep algorithm knowledge
- Gradient Boosted Regression Trees (**GBRT**):
  - → Top-performing HPO algorithm
- **No single best parameter set** or preferred parameter choices
  - → One set per problem description

7. DEMO

SHOW ME!

UNIVERSITÄT
LEIPZIG

THANK YOU!

Daniel Werner
Fakultät für Mathematik und Informatik