Automated Benchmark-Driven Design and Explanation of Hyperparameter Optimizers

Hubert Ruczyński, Anna Kozak

The authors

Bernd Bischl studied computer science, artificial intelligence, and data sciences in Hamburg, Germany; Edinburgh, U.K.; and Dortmund, Germany. He received the Ph.D. degree in statistics from Dortmund Technical University, Dortmund, in 2013 with a thesis on "Model and Algorithm Selection in Statistical Learning and Optimization."

He holds the Chair of Statistical Learning and Data Science with the Department of Statistics, Ludwig-Maximilians-Universität München, Munich, Germany, and he is the Co-Director of the Munich Center for Machine Learning, one of Germany's national competence centers for ML. He is a member of ELLISl, and a Faculty Member of ELLIS Munich, an active developer of several R-packages, leads the "mlr" (Machine Learning in R) Engineering Group and is the Co-Founder of the Science Platform "OpenML" for open and reproducible ML. Furthermore, he leads the Munich branch of the Fraunhofer ADA Lovelace Center for Analytics, Data, and Applications, i.e., a new type of research infrastructure to support businesses in Bavaria, especially in the SME sector. His research interests include AutoML, model selection, interpretable ML, as well as the development of statistical software.

Julia Moosbauer is currently pursuing the Doctoral degree with the Chair for Statistical Learning and Data Science, Ludwig-Maximilians-Universität München, Munich, Germany.

She is a member of the Munich Center for Machine Learning. Her research focus lies in the interface between automated and explainable machine learning with the goal to increase transparency and trust into automated machine learning systems. She is also interested in hyperparameter optimization algorithms (in particular, Bayesian optimization), algorithm configuration, multiobjective optimization, and (sequential) experimental design.

Lars Kotthoff received the Doctoral degree in computer science from the University of St Andrews, St Andrews, U.K., in 2012.

He is an Assistant Professor of Computer Science with the University of Wyoming, Laramie, WY, USA. His research focuses on data-driven combinatorial optimization, automated machine learning, and applying machine learning in other disciplines, for example, materials science.

The authors

Marc Becker received the master's degree in geoinformatics from Friedrich-Schiller University Jena, Jena, Germany, in 2020.

Since 2020, he has been a Research Engineer with the Ludwig Maximilian University of Munich, Munich, Germany, and a Main Developer of the mlr3 optimization packages.

Lennart Schneider is currently pursuing the Doctoral degree with the Chair of Statistical Learning and Data Science, Ludwig-Maximilians-Universität München, Munich, Germany.

His research focuses on automated machine learning, hyperparameter optimization, multiobjective optimization, and neural architecture search.

Michel Lang received the Ph.D. degree in statistics, Dortmund Technical University, Dortmund, Germany, in 2015.

He is a former member of the Munich Center for Machine Learning. He is currently a Scientific Manager of the Research Center Trustworthy Data Science and Security, Dortmund, Germany. He is the author of many popular R packages for machine learning and parallelization, e.g., mlr3 or batchtools. His research areas include machine learning, optimization, and software development.

Florian Pfisterer received the master's degree in statistics from the Ludwig Maximilian University of Munich, Munich, Germany, in 2018, where he is currently pursuing the Ph.D. degree with the Statistical Learning and Data Science Group, Department of Statistics.

His research interests are in the field of automated machine learning, multiobjective optimization, and algorithmic fairness.

Martin Binder is currently pursuing the Doctoral degree with the Chair for Statistical Learning and Data Science, Ludwig-Maximilians-Universität München, Munich, Germany.

He is a member of the Munich Center for Machine Learning. He is mostly working on black-box optimization methods for automatic machine learning and hyperparameter optimization, with a focus on multifidelity optimization. He also works on deep learning, specifically self-supervised learning, for genome sequence classification and analysis.

HPO landscape

"Automated hyperparameter optimization (HPO) has gained great popularity and is an important component of most automated machine learning frameworks. However, the process of designing HPO algorithms is still an unsystematic and manual process: new algorithms are often built on top of prior work, where limitations are identified and improvements are proposed. Even though this approach is guided by expert knowledge, it is still somewhat arbitrary."

Structured HPO

"We present a principled approach to automated benchmark-driven algorithm design applied to multifidelity HPO (MF-HPO). First, we formalize a rich space of MF-HPO candidates that includes, but is not limited to, common existing HPO algorithms and then present a configurable framework covering this space. To find the best candidate automatically and systematically, we follow a programming-by-optimization approach and search over the space of algorithm candidates via Bayesian optimization. We challenge whether the found design choices are necessary or could be replaced by more naive and simpler ones by performing an ablation analysis. "

HPO characteristics

1) Black-Box

"The objective usually provides no analytical information, such as a gradient. Thus, the application of many traditional optimization methods, such as BFGS, is rendered inappropriate or at least questionable."

2) Complex Search Space

"The search space of the optimization problem is often high-dimensional and may contain continuous, integer-valued, and categorical dimensions. Often, there are dependencies between dimensions or even specific hyperparameter values."

3) Expensive

"A single evaluation of the objective function may take hours or days. Thus, the total number of possible function evaluations is often severely limited."

HPO characteristics

4) Low-Fidelity Approximation

"An approximation of the true objective value at lower expense can often be obtained, for example, through a partial evaluation."

5) Low Effective Dimensionality

"The landscape of the objective function can usually be approximated well by a function of a small subset of all dimensions."

HPO Issues

1) The simplicity of an optimization algorithm (i.e., how difficult modifications and extensions are, and on how many dependencies a system relies) heavily influences its adoption in practice.

2) Random search (RS), for example, still enjoys great popularity, as it is extremely simple to implement and parallelize, has almost no overhead, and is able to take advantage of the aforementioned low effective dimensionality.

3) Many multifidelity algorithms, for example, are extensions and further developments of HB that take the fixed successive halving (SH) schedule for granted.

Proposed HPO design

1) Formalization

"We formalize the design space of MF-HPO algorithms and demonstrate that established MF-HPO algorithms represent instances within this space."

2) Framework

"Based on this formalization, we present a rich, configurable framework for MF-HPO algorithms, whose software implementation we call surrogate model-assisted HB (Smashy)."

3) Configuration

"Based on the formalization and framework, we follow an empirical approach to design an MF-HPO algorithm by optimization, given a large benchmark suite. This configuration procedure does not only consider performance but also, e.g., the simplicity of the design."

Proposed HPO design

4) Benchmark

"As in general any HPO algorithm will be applied in a diverse set of application scenarios, we evaluate the performance of our newly designed algorithm on a representative set of problems that were not previously used for its configuration (i.e., a clean test-set approach on the meta-level) and compare them with established implementations of HPO methods."

5) Explanation

"For the resulting MF-HPO system, we systematically assess and explain the effect of different design choices on overall algorithmic performance. Furthermore, we investigate the behavior of algorithmic design components in the context of specific problem scenarios; i.e., we investigate which algorithmic components lead to performance improvements for simple HPO with numeric hyperparameters, AutoML pipeline configuration, and neural architecture search."

Defining the supervised ML task

$\mathcal{D} = ((\mathbf{x}^{(i)}, y^{(i)})) \in (\mathcal{X} \times \mathcal{Y})^n$ of n observations, assumed to be drawn i.i.d. from a data-generating distribution $\mathbb{P}_{xy}$

An ML model is a function $\hat{f} : \mathcal{X} \to \mathbb{R}^g$ that assigns a prediction to a feature vector from $\mathcal{X}$

The inducer $\mathcal{I} : (\mathcal{D}, \boldsymbol{\lambda}) \mapsto \hat{f}$ uses training data $\mathcal{D}$ and a vector of *hyperparameters* $\boldsymbol{\lambda} \in \Lambda$ that govern its behavior

$L : \mathcal{Y} \times \mathbb{R}^g \to \mathbb{R}_0^+$, which is to be minimized during model fitting

The expectation of the loss value of predictions made for data samples drawn from $\mathbb{P}_{xy}$ is the *generalization error*

$$\mathrm{GE} := \mathbb{E}_{(\mathbf{x},y)\sim\mathbb{P}_{xy}} \left[L\left(y, \hat{f}\,(x)\right) \right]$$

Hyperparameter Optimization

$$\boldsymbol{\lambda}_S^* \in \underset{\boldsymbol{\lambda}_S \in \Lambda_S}{\operatorname{argmin}} \; c\left(\boldsymbol{\lambda}_S\right) = \underset{\boldsymbol{\lambda}_S \in \Lambda_S}{\operatorname{argmin}} \; \widehat{\mathrm{GE}}\left(\mathcal{I}, \left(\boldsymbol{\lambda}_S, \boldsymbol{\lambda}_C\right), \mathbf{J}\right)$$

Multifidelity

$$\widehat{\mathrm{GE}}(\mathcal{I}, \boldsymbol{\lambda}, \mathbf{J}) = \frac{1}{N_{\mathrm{iter}}} \sum_{j=1}^{N_{\mathrm{iter}}} L\Big(y[-J_j], \mathcal{I}\big(\mathcal{D}[J_j], \boldsymbol{\lambda}\big)\big(x[-J_j]\big)\Big)$$

$$c(\boldsymbol{\lambda}_S; r) := \widehat{\mathrm{GE}}(\mathcal{I}, (\boldsymbol{\lambda}_S, \boldsymbol{\lambda}_C(r)), \mathbf{J}(r))$$

Generic HPO Algorithm

Algorithm 1 Generic HPO Algorithm

1: **while** budget is not exhausted **do**
2: Propose $\left(\lambda_S^{(i)}, r^{(i)}\right), i = 1, ..., k$, based on archive $\mathcal{A}$
3: Write proposals into a shared archive $\mathcal{A}$
4: Estimate generalization error(s) $c\left(\lambda_S^{(i)}; r^{(i)}\right)$
5: Write results into shared archive $\mathcal{A}$
6: **end while**
7: Wait for workers to synchronize
8: Return best configuration in archive $\mathcal{A}$

Algorithm Configuration

$$\boldsymbol{\gamma}^* \in \operatorname*{argmin}_{\boldsymbol{\gamma}\in\Gamma} \mathbb{E}_{\omega\sim\mathbb{P}_\Omega}\left[\zeta\left(A(\omega,\boldsymbol{\gamma})\right)\right]$$

Formalization of MF-HPO Algorithms

while $t < 1$ **do**

    if $r = 1$ **then** $\triangleright$ Generate new batch of configurations

        $r \leftarrow (\eta_{\text{fid}})^{b-s}$

        $C \leftarrow \textsc{Sample}\big(\mathcal{A}, \mu(b), r; \mathcal{I}_{f_{\text{sur}}}, \mathbb{P}_{\boldsymbol{\lambda}}(\mathcal{A}), \rho(t), \left(N_{\text{s}}^0(t), N_{\text{s}}^1(t)\right), n_{\text{trn}}\big)$

        if *batch_method* $=$ `HB` **then**

            $b \leftarrow (b \bmod s) + 1$

        end if

    else $\triangleright$ Progress fidelity

        $r \leftarrow r \cdot \eta_{\text{fid}}$

        $C \leftarrow \textsc{select\_top}(C, |C|/\eta_{\text{surv}})$

        if *batch_method* $=$ `equal` **then**

            $\tilde{\mu} \leftarrow \mu(b) - |C|$

            $C \leftarrow C \cup \textsc{Sample}\big(\mathcal{A}, \tilde{\mu}, r; \mathcal{I}_{f_{\text{sur}}}, \mathbb{P}_{\boldsymbol{\lambda}}(\mathcal{A}), \rho(t), \left(N_{\text{s}}^0(t), N_{\text{s}}^1(t)\right), n_{\text{trn}}\big)$

        end if

    end if

    Evaluate configuration(s) $c(\boldsymbol{\lambda}_S; r)$ for all $\boldsymbol{\lambda}_S \in C$

    Write results into shared archive $\mathcal{A}$

    $t \leftarrow t + r \cdot |C|/B$ $\triangleright$ Update budget spent

end while

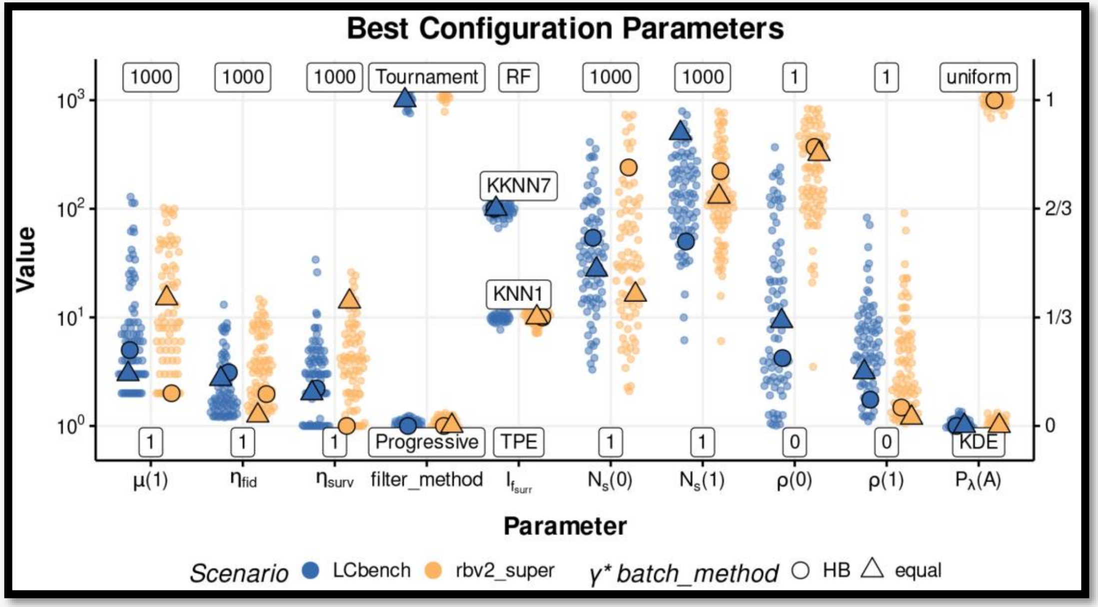
Covered MF-HPOs

Algorithm	$\mu(b)$	s	η_{surv}	η_{budget}	$\mathcal{I}_{f_{\text{sur}}}$	ρ	N_{s}	*batch_mode*	$\mathbb{P}_{\boldsymbol{\lambda}}(\mathcal{A})$
RS	—	1	—	—	—	1	—	—	uniform
BO	1	1	—	—	e.g. GP+EI*	ρ	N_{s}	—	uniform
SH	μ	$\lfloor -\log_\eta(r_{\min}) \rfloor + 1$	η	η	—	1	—	SH	uniform
HB	$\lceil s \cdot \frac{\eta^{s-b}}{s-b+1} \rceil$	$\lfloor -\log_\eta(r_{\min}) \rfloor + 1$	η	η	—	1	—	HB	uniform
BOHB	$\lceil s \cdot \frac{\eta^{s-b}}{s-b+1} \rceil$	$\lfloor -\log_\eta(r_{\min}) \rfloor + 1$	η	η	TPE*	ρ	N_{s}	HB	KDE†

Research Questions

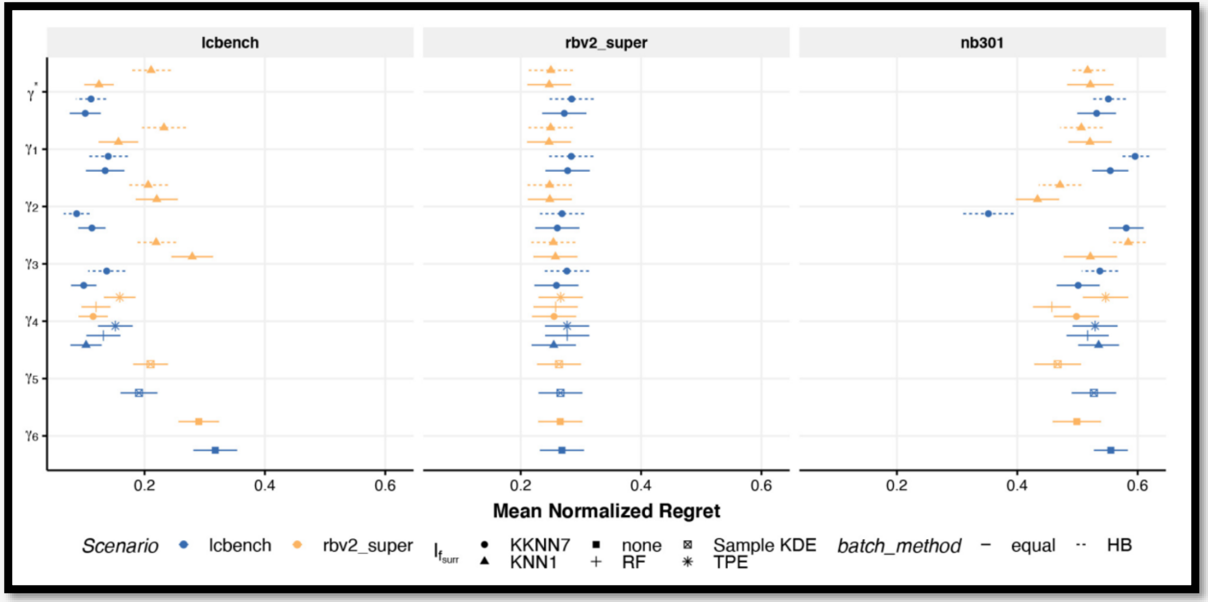
RQ1

How does the optimal configuration differ between problem scenarios, i.e., do different problem scenarios benefit from different HPO algorithms?

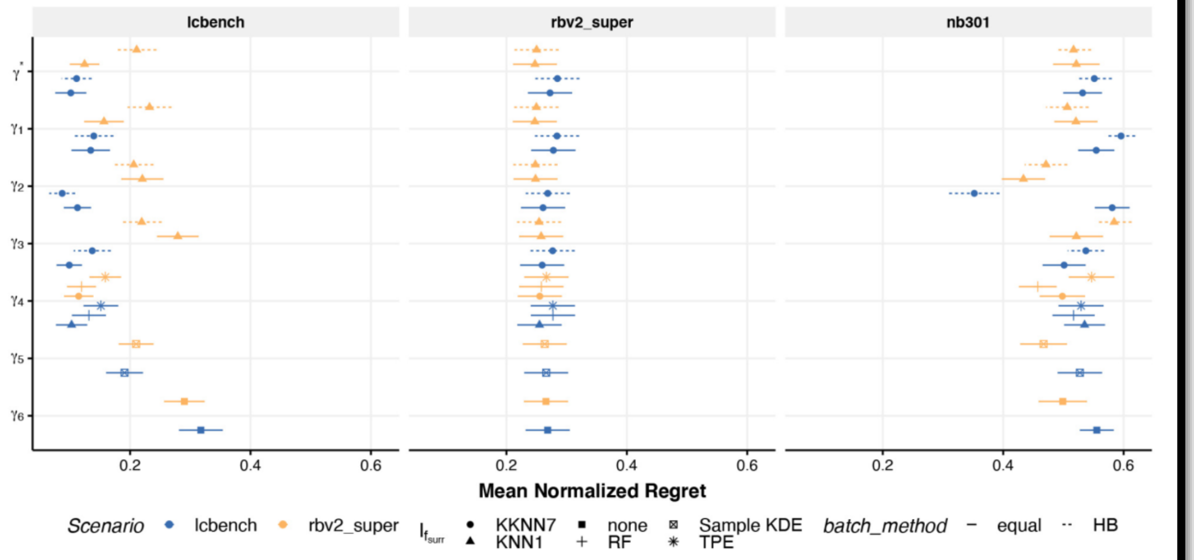


RQ1

How does the optimal configuration differ between problem scenarios, i.e., do different problem scenarios benefit from different HPO algorithms?

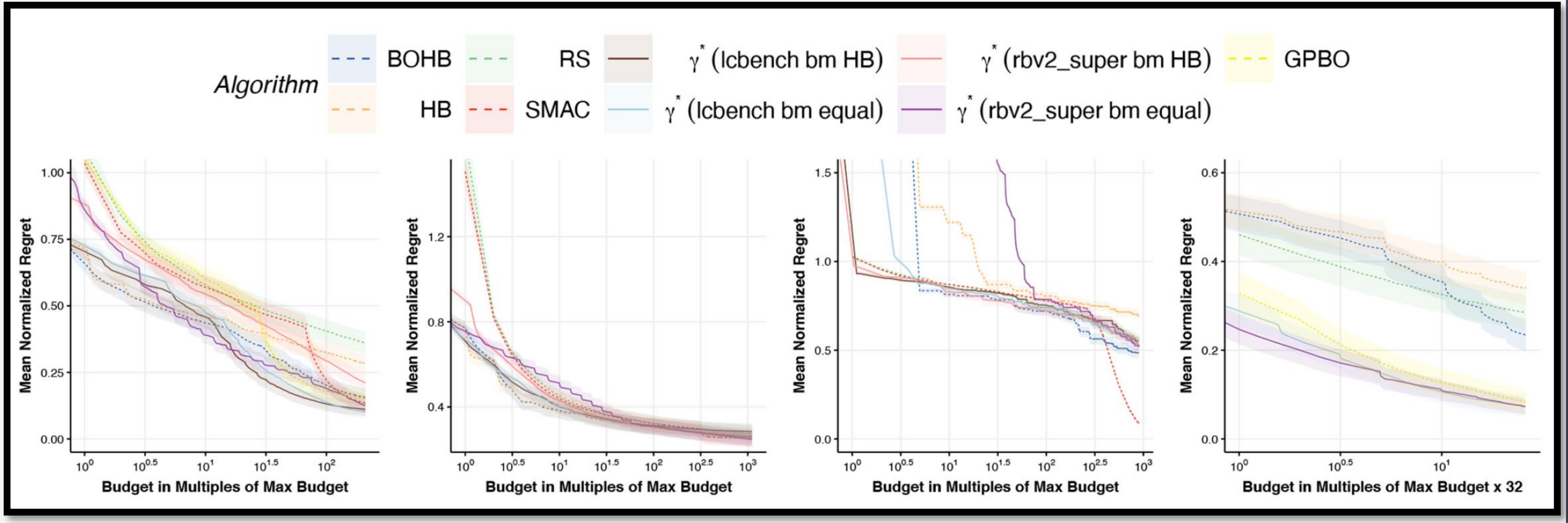


Name	RQ	Optimize	Design Modification
γ^*	1, 2, 3	✓	none (global optimization)
γ_1	4	✗	$\eta_{\mathrm{fid}} \to \infty$
γ_2	5a	✓	$n_{\mathrm{trn}}(0) = n_{\mathrm{trn}}(1),\ N_s^0(0) = N_s^0(1),\ N_s^1(0) = N_s^1(1),\ \rho(0) = \rho(1)$
γ_3	5b	✓	*filter_method*$\to$ `tournament`, $n_{\mathrm{trn}} \to 1,\ N_s^0(0) = N_s^0(1) = N_s^1(0) = N_s^1(1),\ \rho(0) = \rho(1)$
γ_4	6	✗	*batch_method*$\to$ `equal`, $\mathcal{I}_{f_{\mathrm{sur}}} \to *$
γ_5	6	✗	*batch_method*$\to$ `equal`, $\rho \to 0$
γ_6	6	✗	*batch_method*$\to$ `equal`, $\rho \to 0,\ \mathbb{P}_{\boldsymbol{\lambda}}(\mathcal{A}) \to$ `uniform`
γ_7	7	✗	*batch_method*$\to$ `equal`, $\mu \to 32$, quadruple budget



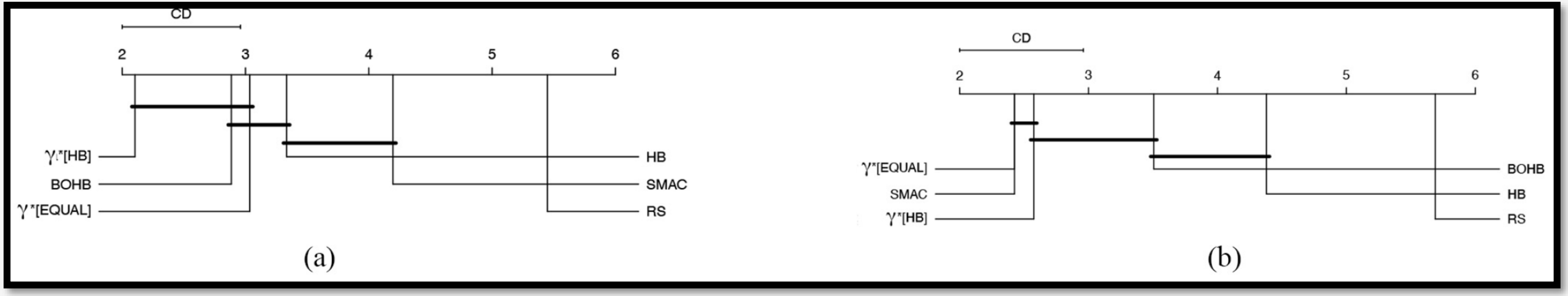
RQ2

How does the optimized algorithm compare to other established HPO implementations?



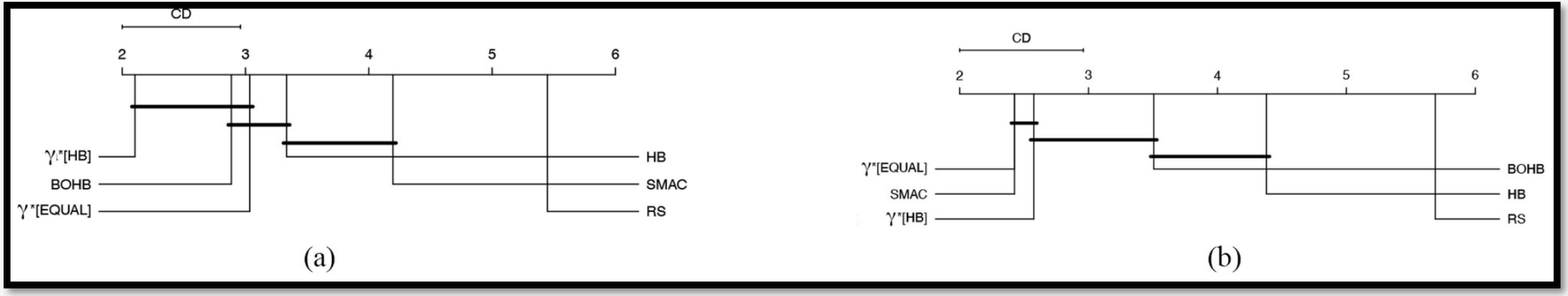
RQ2

How does the optimized algorithm compare to other established HPO implementations?



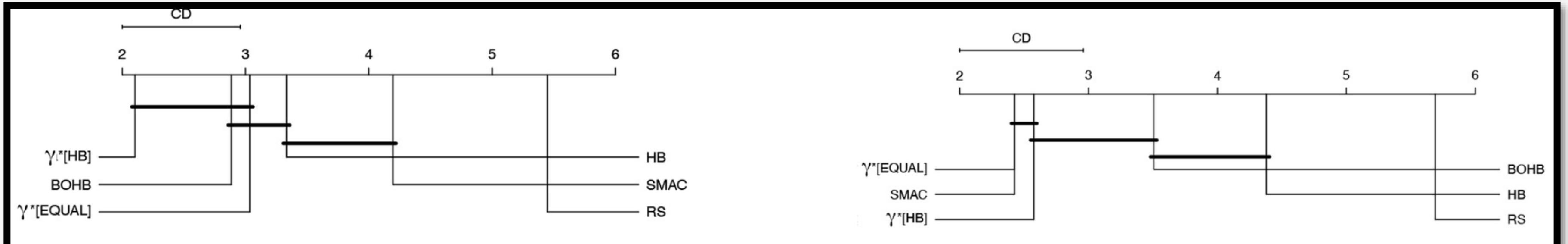
RQ3

Does the successive-halving fidelity schedule have an advantage over the (simpler) equal-batch-size schedule?



RQ4

What is the effect of using multi-fidelity methods in general?



(a) (b)

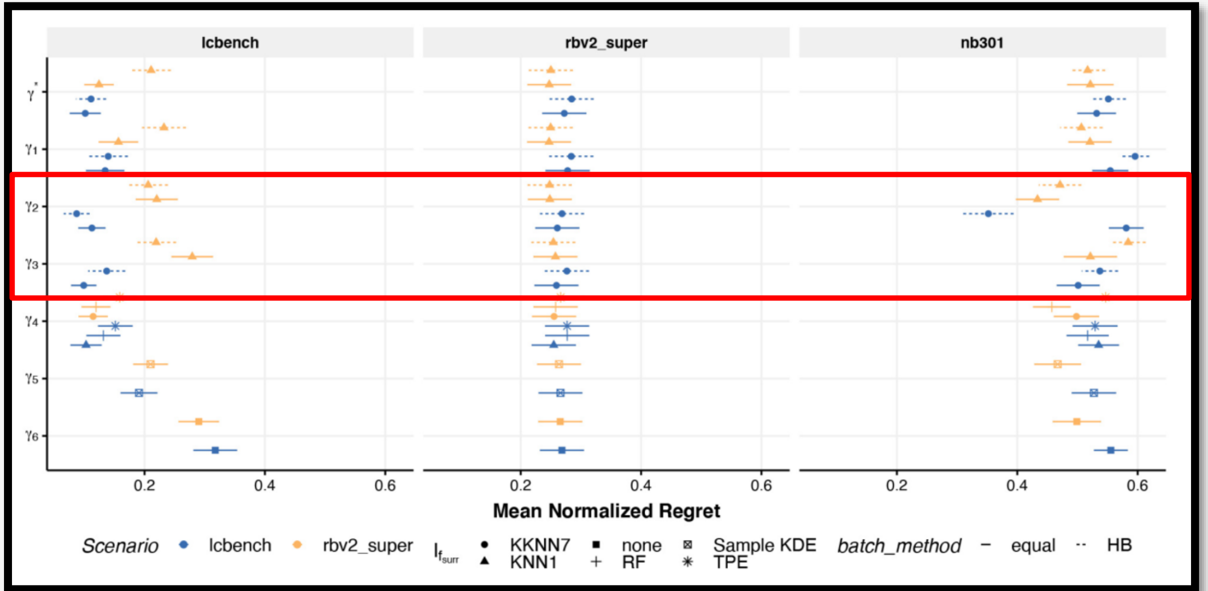
RQ5a and RQ5b

(A) Does changing SAMPLE configuration parameters throughout the optimization process offer an advantage?

(B) Does (more complicated) surrogate-assisted sampling in SAMPLE provide an advantage over using simple random sampling with surrogate filtering?

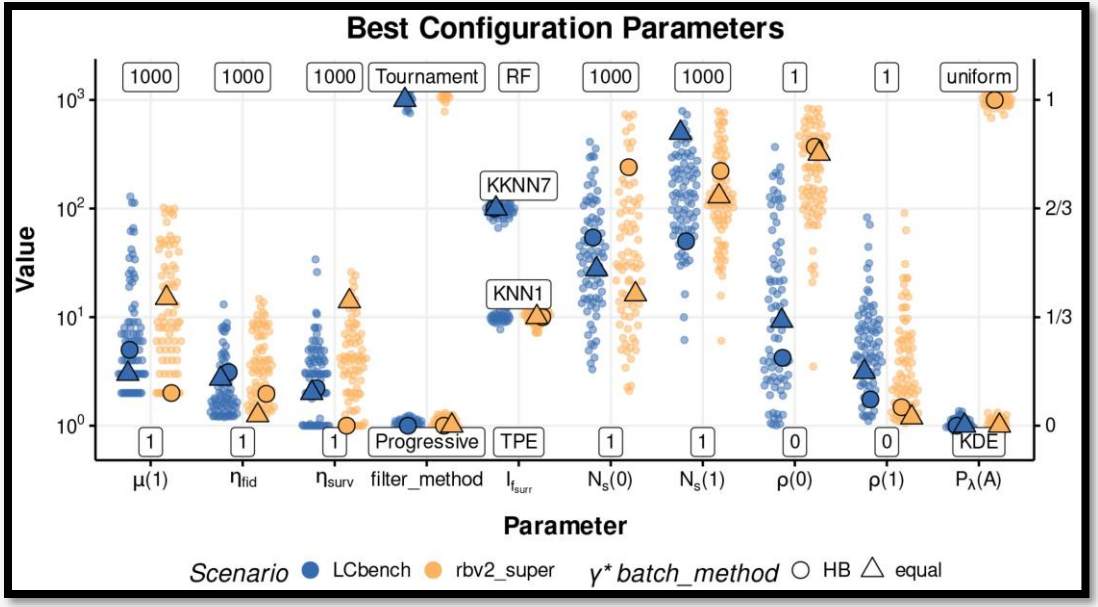
Name	RQ	Optimize	Design Modification
γ^*	1, 2, 3	✓	none (global optimization)
γ_1	4	✗	$\eta_{\text{fid}} \to \infty$
γ_2	5a	✓	$n_{\text{trn}}(0) = n_{\text{trn}}(1),\ N_{\text{s}}^{0}(0) = N_{\text{s}}^{0}(1),\ N_{\text{s}}^{1}(0) = N_{\text{s}}^{1}(1),\ \rho(0) = \rho(1)$
γ_3	5b	✓	*filter_method*→ `tournament`, $n_{\text{trn}} \to 1,\ N_{\text{s}}^{0}(0) = N_{\text{s}}^{0}(1) = N_{\text{s}}^{1}(0) = N_{\text{s}}^{1}(1),\ \rho(0) = \rho(1)$
γ_4	6	✗	*batch_method*→ `equal`, $\mathcal{I}_{f_{\text{sur}}} \to *$
γ_5	6	✗	*batch_method*→ `equal`, $\rho \to 0$
γ_6	6	✗	*batch_method*→ `equal`, $\rho \to 0,\ \mathbb{P}_{\boldsymbol{\lambda}}(\mathcal{A}) \to$ `uniform`
γ_7	7	✗	*batch_method*→ `equal`, $\mu \to 32$, quadruple budget

Name	RQ	Optimize	Design Modification
γ^*	1, 2, 3	✓	none (global optimization)
γ_1	4	✗	$n_{\text{fid}} \to \infty$
γ_2	5a	✓	$n_{\text{trn}}(0) = n_{\text{trn}}(1),\ N_s^0(0) = N_s^0(1),\ N_s^1(0) = N_s^1(1),\ \rho(0) = \rho(1)$
γ_3	5b	✓	*filter_method*$\to$ `tournament`, $n_{\text{trn}} \to 1,\ N_s^0(0) = N_s^0(1) = N_s^1(0) = N_s^1(1),\ \rho(0) = \rho(1)$
γ_4	6	✗	*batch_method*$\to$ `equal`, $\mathcal{I}_{f_{\text{sur}}} \to *$
γ_5	6	✗	*batch_method*$\to$ `equal`, $\rho \to 0$
γ_6	6	✗	*batch_method*$\to$ `equal`, $\rho \to 0,\ \mathbb{P}_{\boldsymbol{\lambda}}(\mathcal{A}) \to$ `uniform`
γ_7	7	✗	*batch_method*$\to$ `equal`, $\mu \to 32$, quadruple budget



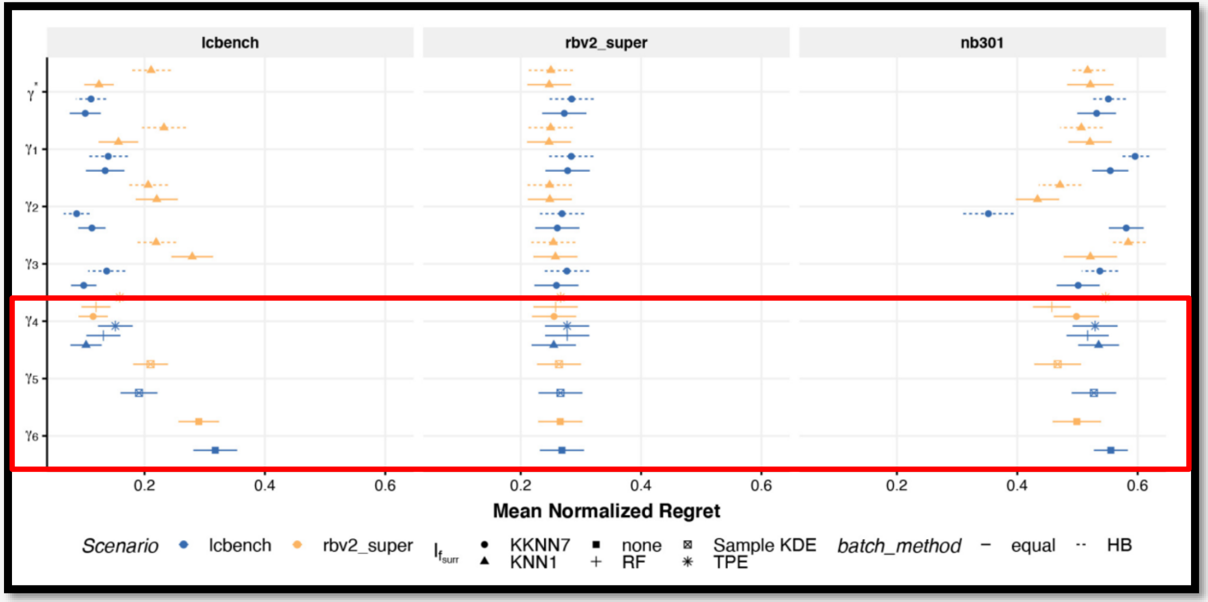
RQ6

What effect do different surrogate models (or using no model at all) have on performance?



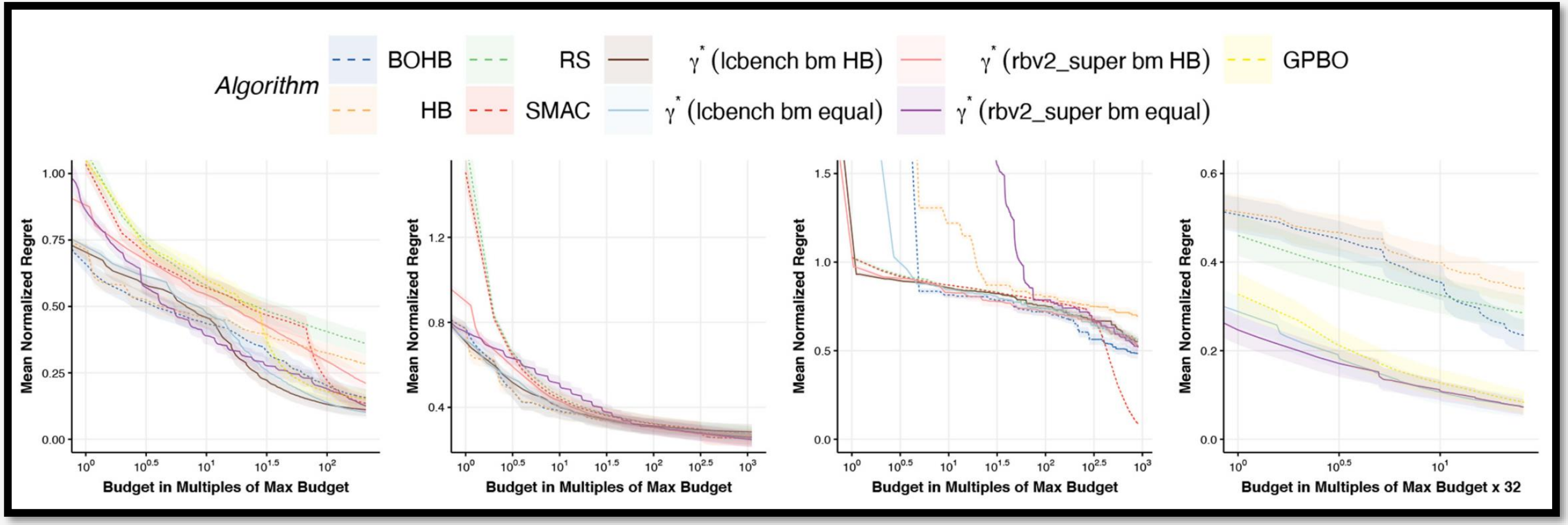
RQ6

What effect do different surrogate models (or using no model at all) have on performance?



RQ7

Does the equal-batch-size schedule give an advantage over established methods when parallel resources are available?



Conclusions