# Facilitating Database Tuning with Hyper-Parameter Optimization: A Comprehensive Experimental Evaluation

[Experiment, Analysis & Benchmark]

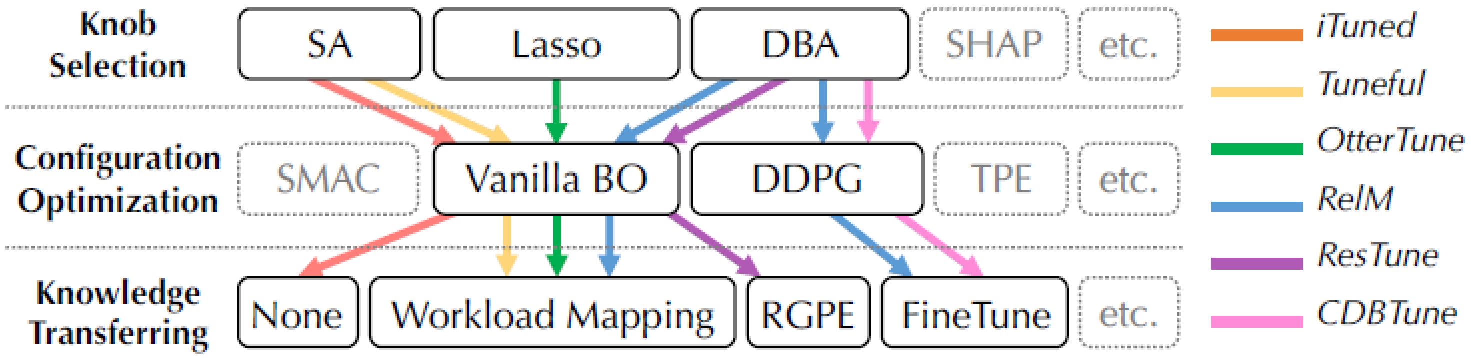
[2023년 3월] 연세대학교 컴퓨터과학과 이지은

과제명 : IoT 환경을 위한 고성능 플래시 메모리 스토리지 기반 인메모리 분산 DBMS 연구개발

과제번호: 2017-0-00477

# 1 Introduction

- It remains a challenge to select the best solution for database configuration tuning, considering the large body of algorithm choices.
- We could find more potential algorithms designed for configuration tuning.
- We summarize three key modules of database configuration tuning systems and conduct extensive ablation studies using various challenging cases.
- Three key modules in the existing tuning systems: knob selection, configuration optimization, knowledge transfer



# 1.1 Motivation

I1. Missing comparative evaluations of intra-algorithms in different modules.

Existing researches focus on the evaluation of the entire tuning systems or limited intra-algorithms in some of the modules, failing to reveal which intra-algorithm contributes to the overall success

I2. Absence of analysis for high-dimensional and heterogeneous scenarios.

DBMS has hundreds of configuration knobs that could be continuous or categorical ➔ heterogeneity

I3. Limited solution comparison without a broader view beyond database community.

Such opportunities to further facilitate database configuration tuning have not been investigated in the literature

I4. Lack of cheap-to-evaluate and unified database tuning benchmarks.

Evaluating new algorithms in database tuning systems can be costly, time-consuming, and hard to interpret

# 1.2 Our Contributions

C1. We present a unified pipeline with three key modules and evaluate the fine-grained intra-algorithms. (For I1)

(1) How to determine tuning knobs?
(2) Which optimizer is the winner?
(3) Can we transfer knowledge to speed up the target tuning task?

C2. We construct extensive scenarios to benchmark multiple optimizers. (For I2)

For high dimensionality, we conduct evaluations of the optimizers over configuration spaces of three sizes: small, medium, and large

C3. Think out of the box: we apply and evaluate advanced HPO techniques in database tuning problems. (For I3)

Such opportunities to further facilitate database configuration tuning have not been investigated in the literature

C4. We define an efficient database configuration tuning benchmark via surrogates. (For I4)

We train regression models on (configuration, performance) pairs collected in an expensive offline manner and cheaply evaluate future configurations using the model's performance predictions instead of replying workloads

# 2.3 Taxonomy

## 2.3.1 BO-based

(1) iTuned
(2) OtterTune
(3) Tuneful
(4) ResTune
(5) RelM
(6) CGPTuner

## 2.3.2 RL-based

(1) CDBTune
(2) QTune

**Table 1: Taxonomy and Brief Description of Existing Database Configuration Tuning Systems.**

| Category | Configuration Tuning System | Application | Design Highlight |
|---|---|---|---|
| BO-based | iTuned [19] | Performance tuning for DBMS | First adopting BO |
| | OtterTune [4] | Performance tuning for DBMS | Incremental knob selection, workload mapping |
| | Tuneful [25] | Performance tuning for analytics engines | Incremental knob selection via Sensitivity Analysis |
| | ResTune [94] | Resourse-oriented tuning for DBMS | Adopting RGPE to transfer historical knowledge |
| | RelM [46] | Memory allocation for analytics engines | Combining white-box knowledge |
| | CGPTuner [15] | Performance tuning for IT systems | Adopting Contextual BO to adapt to workload variation |
| RL-based | CDBTune [93] | Performance tuning for DBMS | First adopting DDPG |
| | QTune [49] | Performance tuning for DBMS | Supporting three tuning granularities |

# 3.1 Knob Selection

**3.1.1 Variance-based Measurements** select the knobs that have the largest impact on the database performance

(1) Lasso

Based on coefficient of linear regression, effective for when existing irrelevant features

(2) Gini Score

Based on the times a feature is used in tree splits, successful in high-dimensional feature selection

(3) fANOVA (functional analysis of variance)

Decomposing the variance of the target function, commonly used in the HPO field

Measureing the importance of knobs by analyzing how much each knob contributes to the variance of the target function across the configuration space

Based on a regression model, functional ANOVA decomposed the target function into additive components that only depend on subsets of its inputs

**3.1.2 Tunability-based Measurements** quantifies the tunability of a knob, measuring the performance gain that can be achieved by tuning the knob from its default value

(1) Ablation Analysis

Comparing feature difference between configuration, straightforward and intuitive

Selecting the knob whose change contributes the most to improve the performance of configurations

For speedup, the evaluations are replaced by cheap prediction obtained from surrogates

(2) SHAP (SHapley Additive exPlanations)

Decomposing the performance change additively, solid theoretical foundation derived from game theory

SHAP computes each knob’s contribution(i.e., SHAP value) for pushing the default performance to the target one

# 3.2 Configuration Optimization

(1) Vanilla BO

The BO-based optimizer that adopts vanilla GP as its surrogate model.

(2) One-Hot BO

The BO-based optimizer that adopts one-hot encoding for categorical variables, as original GPs assume continuous input variables.
Specifically, each categorical feature with *k* possible values is converted into *k* binary features.

(3) Mixed-Kernel BO (2017. ROBO : A Flexible and Robust Bayesian Optimization Framework in Python)

The BO-based optimizer that adopts a mixed-kernel GP as its surrogate model.
→ Matern kernel for continuous knobs
→ Hamming kernel for categorical knobs (also one-hot encoded), and calculates their product

(4) SMAC (Sequential Model-based Algorithm Configuration)

Adopting a random forest based surrogate
Assuming a Gaussian model $N(y|\hat{\mu}, \hat{\sigma}^2)$, where $\hat{\mu}$ and $\hat{\sigma}^2$ are the mean and variance of the random forest.
Supporting all types of variables, including continuous, discrete, and categorical features.

# 3.2 Configuration Optimization

(5) TPE (Tree-structured Parzen estimator)

A non-standard Bayesian optimization algorithm.

While GP and SMAC modeling the probability $p(y|\theta, H)$ directly, TPE models $p(\theta|y, H)$ by tree-structured Parzen density estimators.

TPE describes the configuration space by a generative process and supports categorical features.

(6) TuRBO (Trust-Region BO)

A local strategy for global optimization using independent surrogate models

(7) DDPG (Deep Deterministic Policy Gradient algorithm)

While deep-Q learning are limited to setting a knob from a finite set of predefined values, DDPG can work over a continuous action space, setting a knob to any value within a range.

Consisted of the neural networks: actor and critic

→ The actor decides how to suggest a configuration (choosing an action based on the input states)

→ The critic provides feedback on the suggestion to guide the actor (evaluating the selected action based on the reward)

(8) GA (Genetic Algorithm)

Meta-heuristic inspired by the process of natural selection.

In each iteration, the fitness of each solution, which is usually the value of the object

| Algorithm | High-dimensionality | Heterogeneity |
|---|---|---|
| Vanilla BO | – | – |
| One-Hot BO | – | ✓ |
| Mixed-Kernel BO [41] | – | ✓ |
| SMAC [35] | ✓ | ✓ |
| TPE [10] | – | ✓ |
| TurBO [23] | ✓ | – |
| DDPG [54] | ✓ | – |
| GA [48] | – | ✓ |

# 3.3 Knowledge Transfer

The knowledge transfer module is designed to accelerate the target tuning task by leveraging the experience from historical tuning tasks

(1) Workload Mapping (proposed by OtterTune)

It matches the target workload to the most similar historical one based on the absolute distances of database metrics and reuses the historical observations from the similar workload.

(2) RGPE (ranking-weighted Gaussian process ensemble)

An ensemble mode for BO-based optimziers.
RGPE combined similar base GP models of historical tasks via distinguishable weights.
The weights are assigned using relative ranking loss to generalize across different workloads and various hardware environments.

(3) Fine-tune

Used in the RL-based optimizers.
For example, CDBTune and Qtune could pre-train a basic DDPG model by replaying historical workload.
It helps the optimizer adapt to different workloads with fewer observations instead of training from scratch.

# 5 HOW TO DETERMINE TUNING KNOBS?

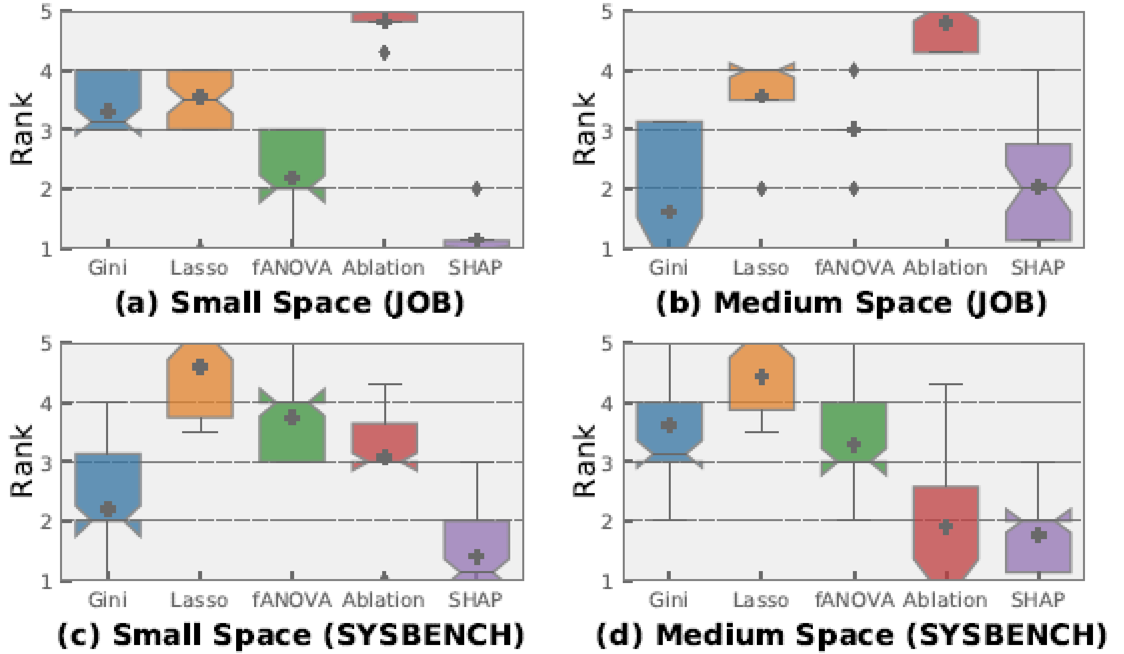



Figure 3: Performance ranking of importance measurements. (Notch denotes medium and plus sign denote mean.)

Table 5: Overall performance ranking (bold values are best).

| Measurement | Gini | Lasso | fANOVA | Ablation Analysis | SHAP |
|---|---|---|---|---|---|
| Average Ranking | 2.62 | 4.18 | 3.06 | 3.45 | 1.67 |

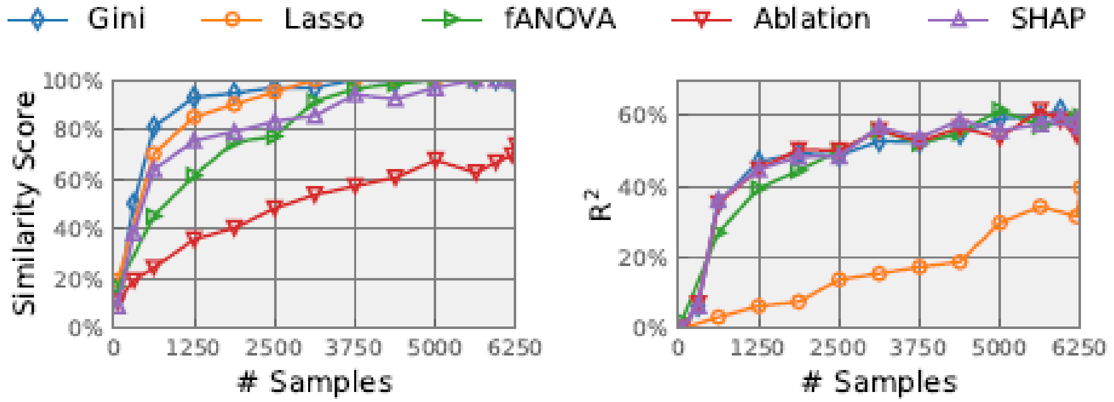



Figure 4: Sensitivity analysis for importance measurements.

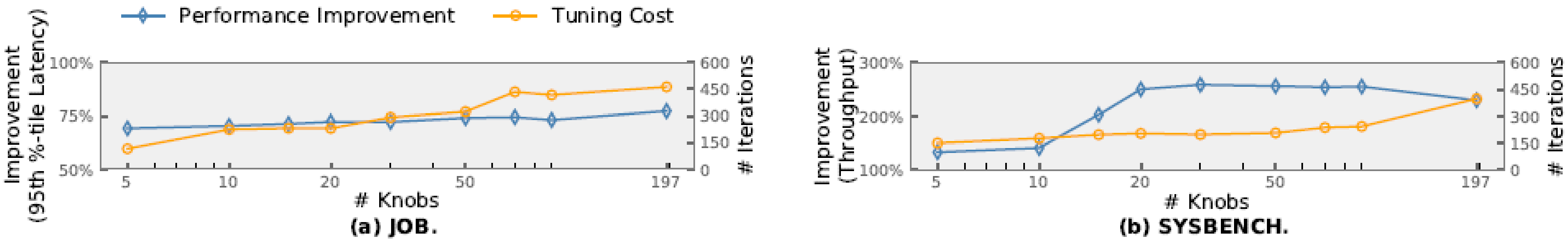



Figure 5: Performance improvement and tuning cost when increasing the number of tuning knobs.

# 5.4 Main Findings

- Given a limited tuning budget, tuning over the configuration space with all the knobs is inefficient. It is recommended to preselect important knobs to prune the configuration space.

- Configuration spaces determined by different importance measurements will impact tuning performance significantly.

- **SHAP is the best importance measurement** based on our evaluation. Compared with traditional measurements (i.e., Lasso and Gini score), it achieves 38.02% average performance improvement. When training samples are limited, Gini score is also effective.

- When determining the number of tuning knobs, there is a tradeoff between the performance improvement and tuning cost. Increasing/decreasing the number of tuning knobs has good performances in some cases. However, how to determine the number theoretically is still an open problem with research opportunities.

# 6 WHICH OPTIMIZER IS THE WINNER?

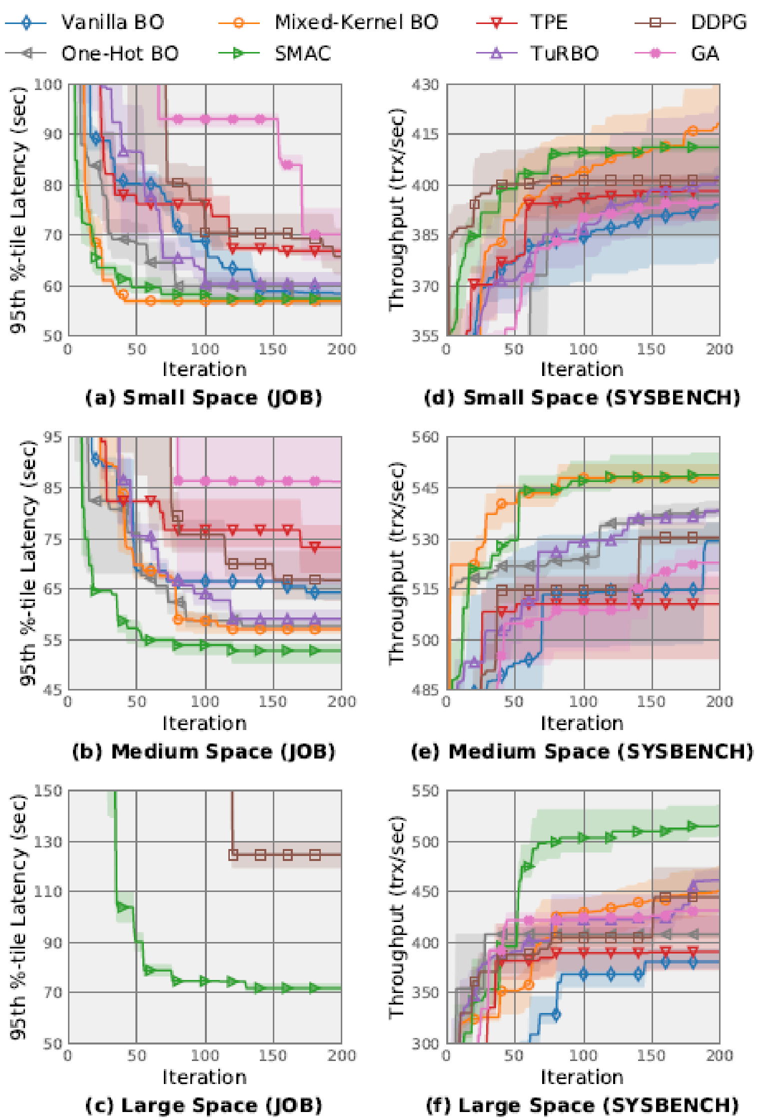



Figure 7: Best performance of optimizers over iteration (for JOB, bottom left is better; for SYSBENCH, top left is better).

Table 6: Average ranking of optimizers in terms of the best performance. VBO, OHBO, MBO denote vanilla BO, one-hot BO, mixed-kernel BO respectively. (bold values are the best.)

| Optimizer | VBO | OHBO | MBO | SMAC | TPE | TuRBO | DDPG | GA |
|---|---|---|---|---|---|---|---|---|
| Small | 5.33 | 4.00 | **2.17** | 3.33 | 5.83 | 3.83 | 5.00 | 6.50 |
| Medium | 5.17 | 3.83 | 2.33 | **1.33** | 7.17 | 4.00 | 5.50 | 6.67 |
| Large | 7.33 | 6.50 | 5.17 | **1.00** | 6.50 | 5.00 | 3.17 | 5.83 |
| **Overall** | 5.94 | 4.78 | 3.22 | **1.89** | 6.50 | 4.28 | 4.56 | 6.33 |

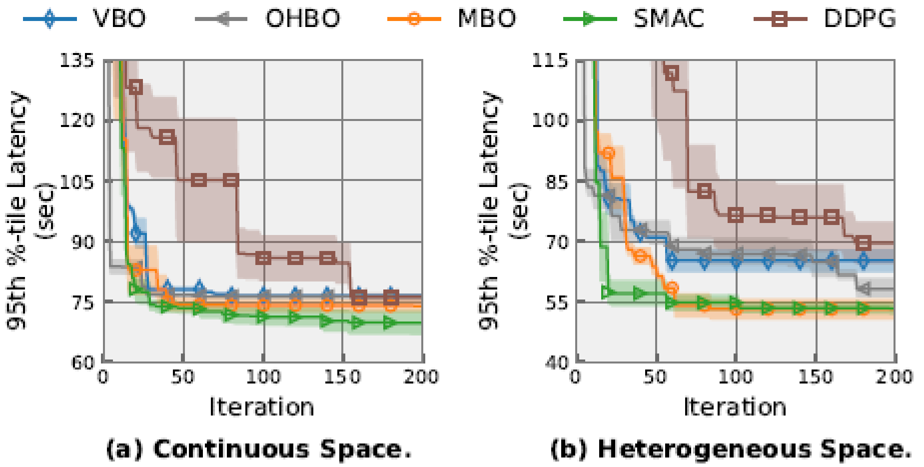



Figure 8: Comparison experiment for knobs heterogeneity (bottom left is better).

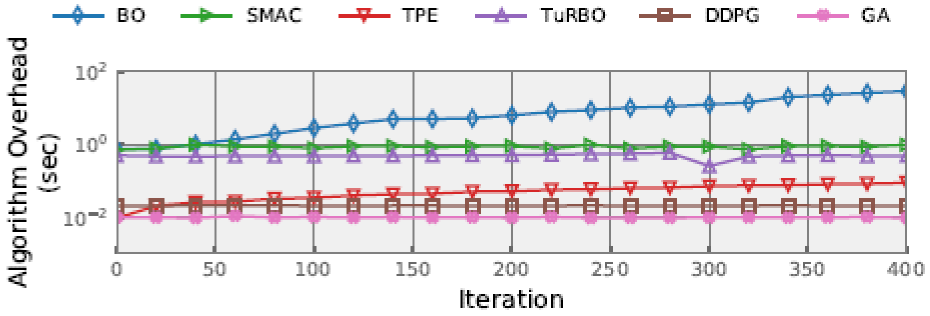



Figure 9: Algorithm overhead for different optimizers.

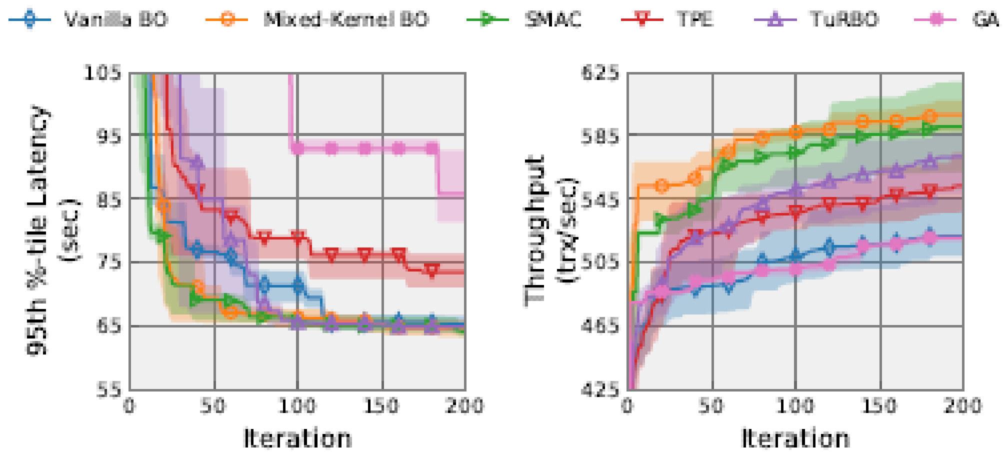
# 6.4 Main Findings

- **SMAC has the best overall performance** and could simultaneously handle the high-dimensionality and heterogeneity of configuration space. Compared with traditional optimizer (i.e., vanilla BO, DDPG), it achieves 21.17% average performance improvement.
- TPE is worse than other optimizers in most cases since it struggles to model the interaction between knobs.
- DDPG has considerable tuning costs (i.e., more iterations) in small and medium configuration spaces due to its redundant MDP modeling and complexity of the neural network. Meanwhile, it has a relatively good performance in a large configuration space.
- On small and medium configuration spaces, SMAC and Mixed-kernel BO rank the top two, while on the large configuration space, SMAC, DDPG, and TuRBO all have good performance rankings. The effectiveness of global GP methods decreases as the number of tuning knobs increases.
- Mixed-kernel BO outperforms other BO-based optimizers in heterogeneous space due to its Hamming kernel measurement.
- Global GP-based optimizers (i.e., vanilla BO, one-hot BO and mixed-kernel BO) have cubic algorithm overhead.

# 7 CAN WE TRANSFER KNOWLEDGE TO SPEED UP THE TARGET TUNING TASK?

**Table 7: Evaluation results for different transfer frameworks (the bold values are the best). We report speedup, performance enhancement (i.e., PE) against the base optimizers and the absolute performance ranking (i.e., APR).**

| Transfer Framework | RGPE | | | | | | Workload Mapping | | | | | | Fine-Tune | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Base Optimizer | Mixed-Kernel BO | | | SMAC | | | Mixed-Kernel BO | | | SMAC | | | DDPG | | |
| Metric | Speedup | PE | APR | Speedup | PE | APR | Speedup | PE | APR | Speedup | PE | APR | Speedup | PE | APR |
| TPCC | **98.28** | **10.44%** | **1** | 8.03 | 2.18% | 2 | × | -2.48% | 4 | 0.35 | 2.43% | 3 | 1.71 | 3.75% | 4 |
| SYSBENCH | **4.76** | 0.53% | 4 | 0.78 | **13.32%** | **1** | × | -0.51% | 5 | 3.08 | 2.23% | 2 | 0.93 | 4.59% | 3 |
| Twitter | **28.42** | 1.56% | **1** | **28.42** | 0.02% | 2 | × | -1.70% | 5 | × | -0.12% | 4 | 0.83 | **3.12%** | 3 |
| **Avg.** | **51.52** | 4.18% | 2 | 12.41 | **5.17%** | **1.67** | × | -1.56% | 4.67 | 1.14 | 1.51% | 3.00 | 1.32 | 3.82% | 3.33 |



(a) JOB. (b) SYSBENCH.

Figure 10: Tuning Performance over surrogate benchmark.

**Table 8: Regression performance (bold values are the best).**

| Model | RF | | GB | | SVR | | NuSVR | | KNN | | RR | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Metric | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ | RMSE | $R^2$ |
| SYSBENCH | **26.5** | **93.0%** | 27.2 | 92.6% | 97.4 | 5.6% | 97.4 | 5.6% | 54.6 | 70.2% | 64.1 | 59.1% |
| JOB | 11.8 | 97.4% | **11.1** | **97.7%** | 41.7 | 67.9% | 41.7 | 67.9% | 27.5 | 86.0% | 52.3 | 49.5% |

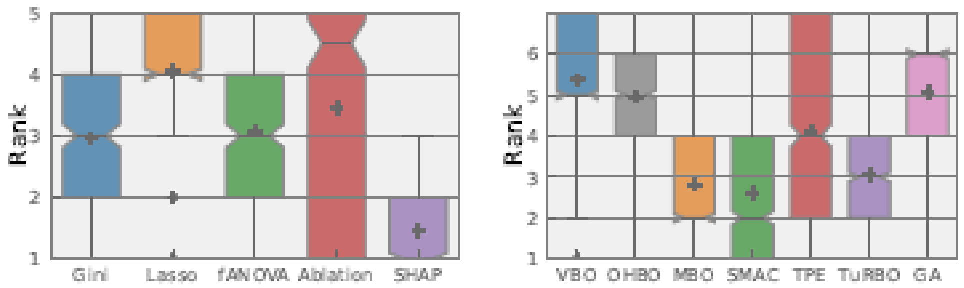



Figure 11: Performance ranking of importance measurements and optimizers evaluated by the tuning benchmark. (Notch denotes medium and plus sign denote mean.)