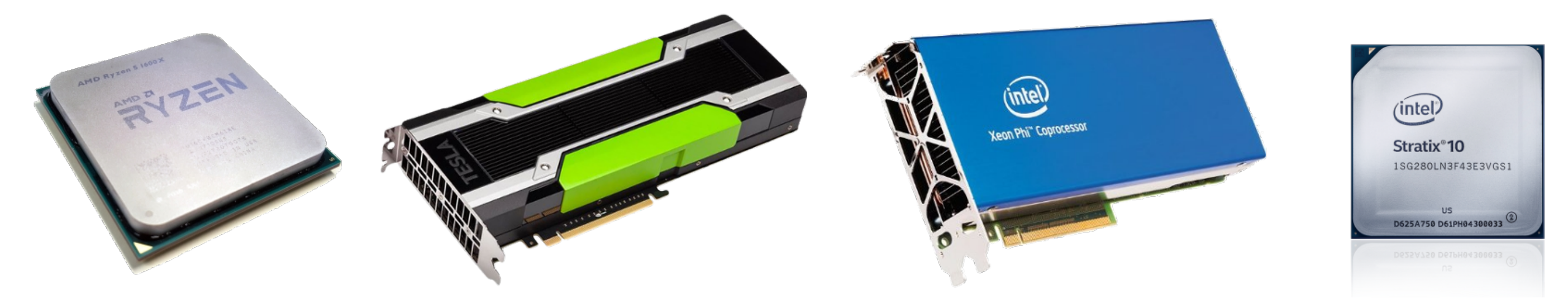
# Autotuning under Tight Budget Constraints: A Transparent Design of Experiments Approach

Pedro Bruel, Steven Quinito Masnada, Brice Videau, Arnaud Legrand, Jean-Marc Vincent, Alfredo Goldman

POLARIS - Systèmes Répartis, Calcul Parallèle et Réseaux

## Autotuning: Optimizing Program Configurations



- How to write efficient code for each of these?
- We can use autotuning: the process of automatically finding a configuration of a program that optimizes an objective

## Strategies for Exploring Search Spaces

| System | Domain | Approach |
|---|---|---|
| ATLAS | Dense Linear Algebra | Exhaustive |
| INSIEME | Compiler | Genetic Algorithm |
| Active Harmony | Runtime | Nelder-Mead |
| ParamILS | Domain-Agnostic | Stochastic Local Search |
| OPAL | Domain-Agnostic | Direct Search |
| OpenTuner | Domain-Agnostic | Ensemble |
| MILEPOST GCC | Compiler | Machine Learning |
| Apollo | GPU kernels | Decision Trees |

Exhaustive, Meta-Heuristics, Machine Learning

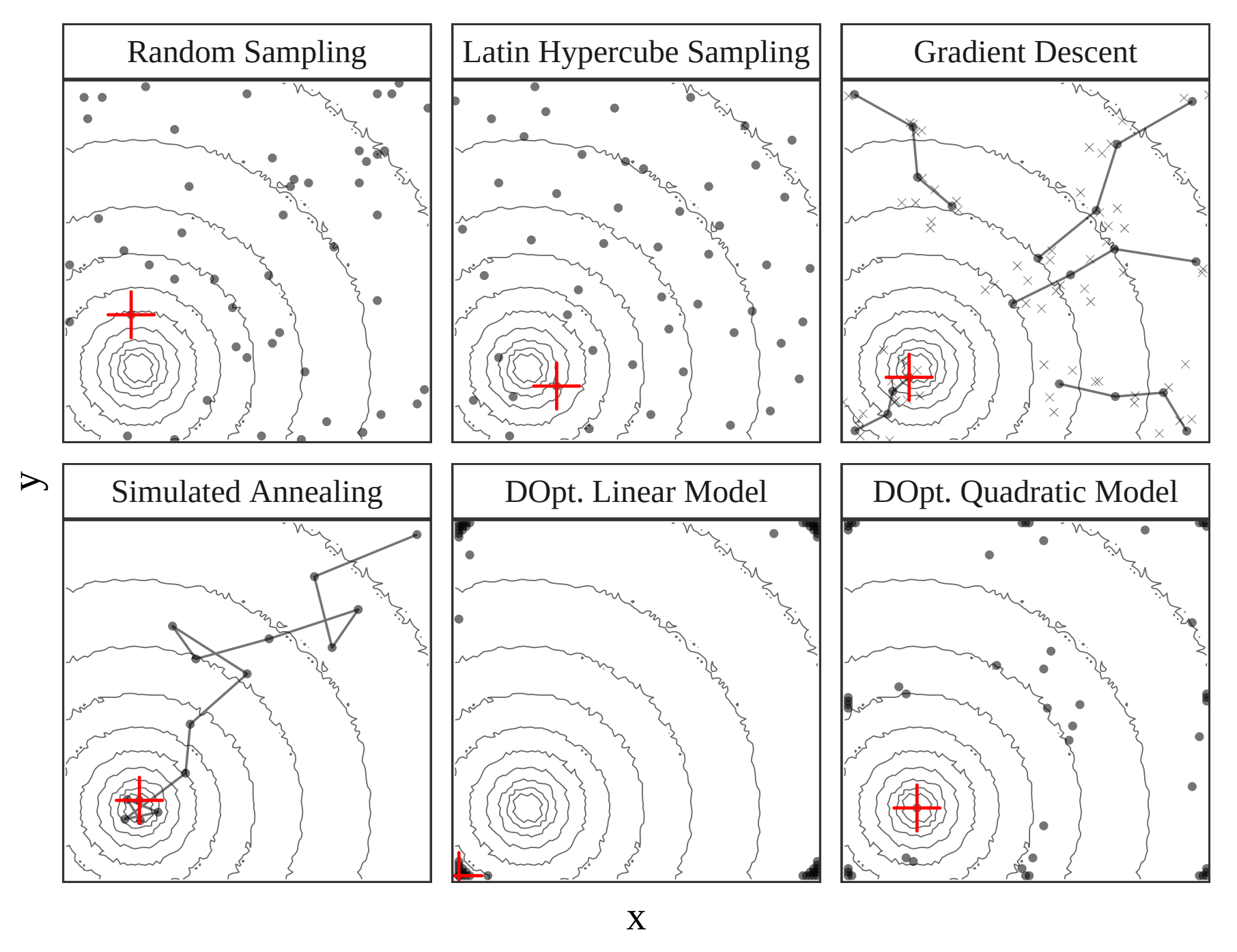
- These approaches need a large number of function evaluations, assuming seach space "smoothness", and that good solutions are reachable
- After optimizing, we learn "nothing" about the search space, and can't explain why optimizations work

## Design of Experiments: Exploration under a Budget

| Run | A | B | C | D | E | F | G |
|---|---|---|---|---|---|---|---|
| 1 | 1 | -1 | 1 | -1 | -1 | 1 | 1 |
| 2 | 1 | 1 | 1 | -1 | 1 | -1 | -1 |
| 3 | -1 | 1 | -1 | -1 | 1 | 1 | 1 |
| 4 | -1 | 1 | 1 | 1 | -1 | 1 | -1 |
| 5 | 1 | -1 | -1 | 1 | 1 | 1 | -1 |
| 6 | 1 | 1 | -1 | 1 | -1 | -1 | 1 |
| 7 | -1 | -1 | 1 | 1 | 1 | -1 | 1 |
| 8 | -1 | -1 | -1 | -1 | -1 | -1 | -1 |

A Plackett-Burman design for 7 2-level factors

- Experiment results can be used to identify relevant parameters and to fit a linear regression model



- Exploration of a search space using a fixed budget of 50 points, the red "+" represents the best point found by each strategy

## A Motivating Result on a GPU Kernel

- Kernel factors:

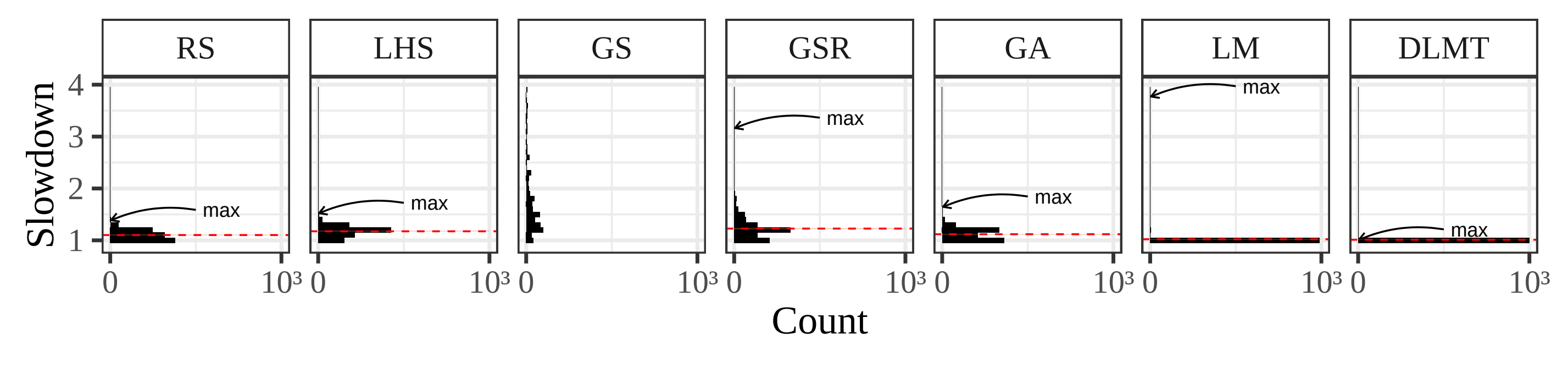
| Factor | Levels | Short Description |
|---|---|---|
| vector_length | $2^0, \dots, 2^4$ | Size of support arrays |
| load_overlap | *true, false* | Load overlaps in vectorization |
| temporary_size | $2, 4$ | Byte size of temporary data |
| elements_number | $1, \dots, 24$ | Size of equal data splits |
| y_component_number | $1, \dots, 6$ | Loop tile size |
| threads_number | $2^5, \dots, 2^{10}$ | Size of thread groups |
| lws_y | $2^0, \dots, 2^{10}$ | Block size in $y$ dimension |

- Initial performance model:

$$\begin{aligned}\text{time\_per\_pixel} \sim\ & \text{y\_component\_number} + \frac{1}{\text{y\_component\_number}} + \\ & \text{load\_overlap} + \text{temporary\_size} + \\ & \text{vector\_length} + \text{lws\_y} + \frac{1}{\text{lws\_y}} + \\ & \text{elements\_number} + \text{threads\_number} + \\ & \frac{1}{\text{elements\_number}} + \frac{1}{\text{threads\_number}}.\end{aligned}$$

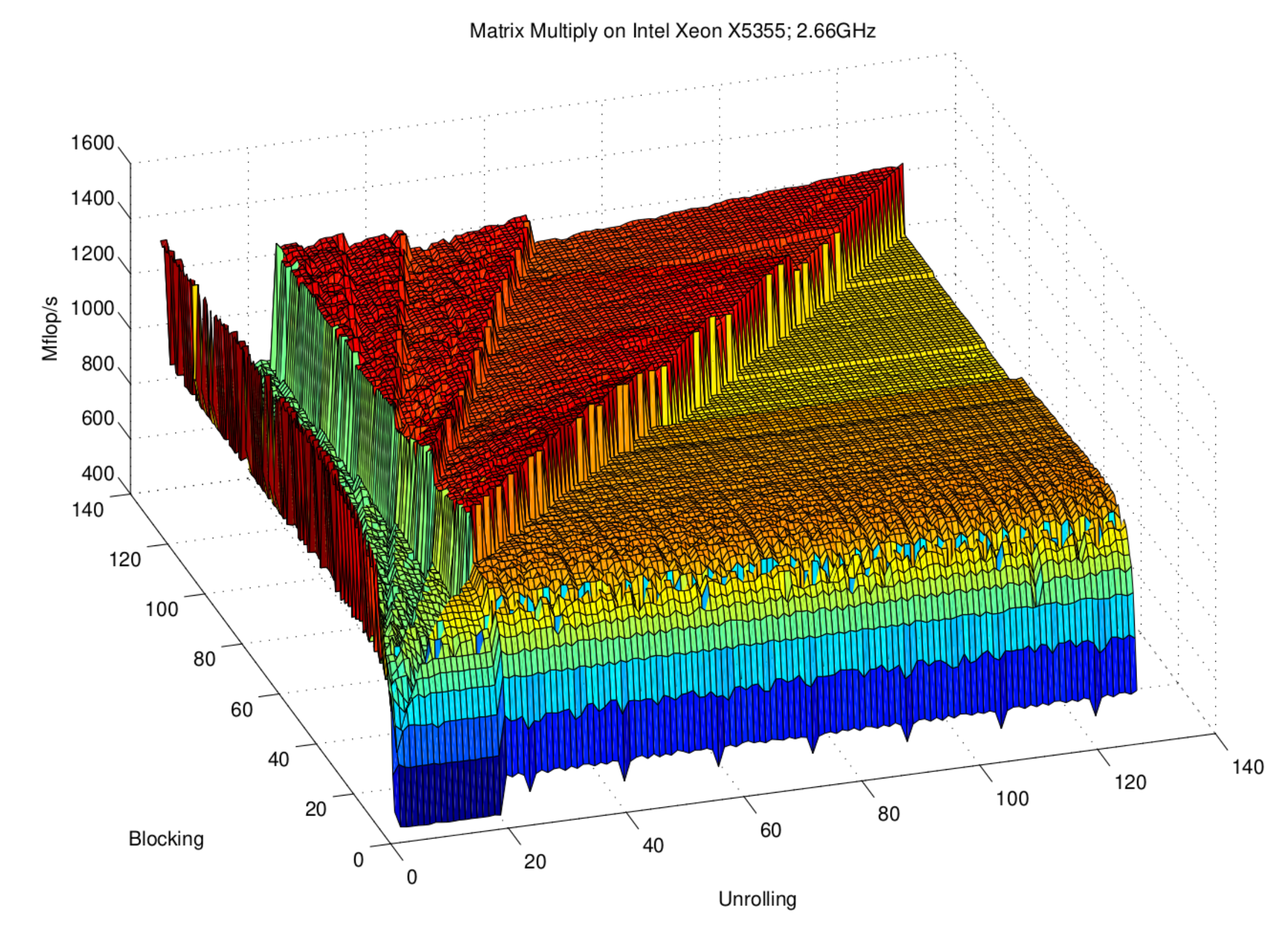
- This simple case had known valid search space and global optimum, and fixed budget

Our approach (DLMT) was always within 1% of the optimum



| RS | LHS | GS | GSR | GA | LM | DLMT |
|---|---|---|---|---|---|---|
| Random Sampling | Latin Hyper Square | Greedy Search | Greedy with Restart | Genetic Algorithm | Linear Model | Our DoE Approach |

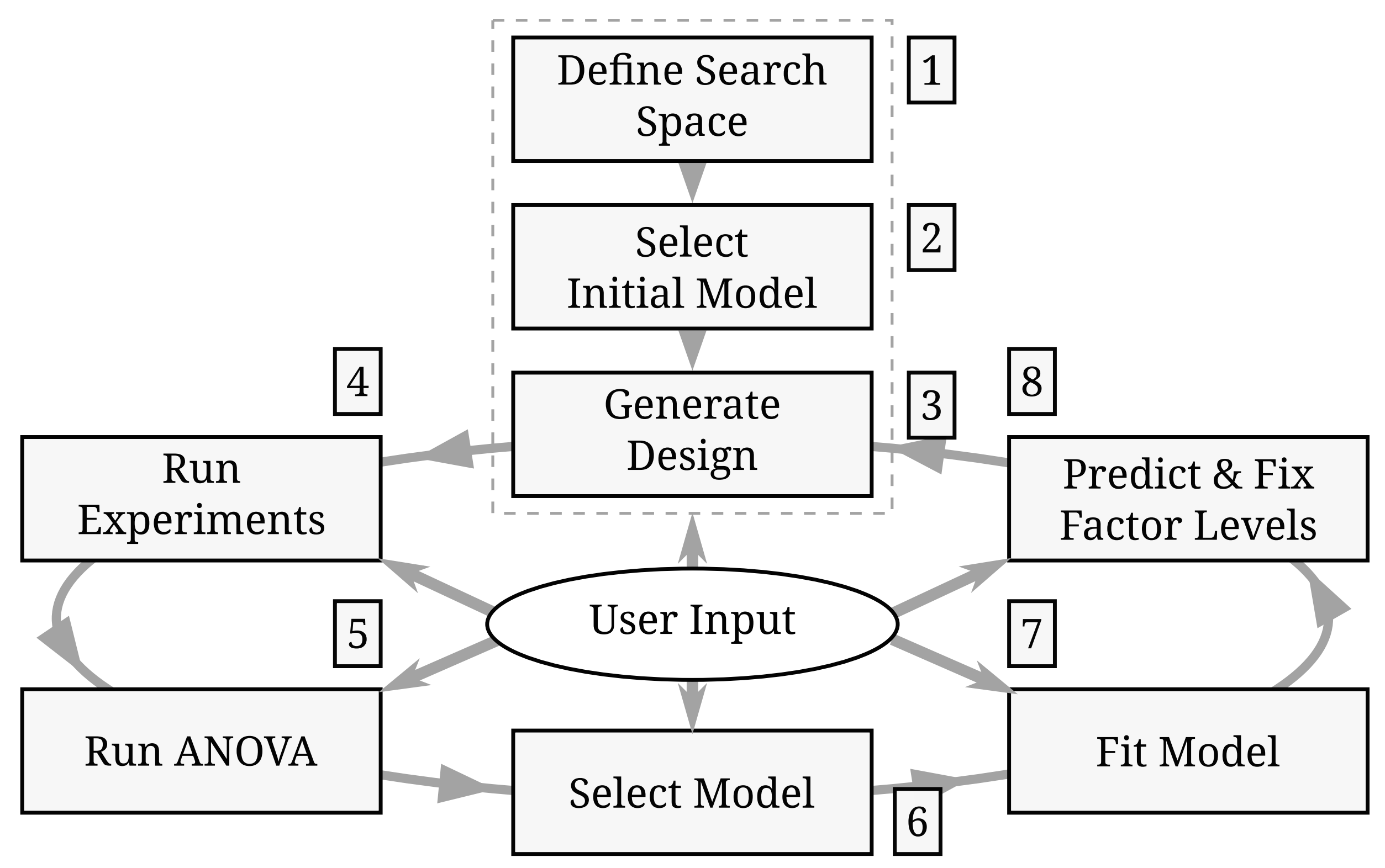
## Autotuning: Search Spaces are Hard to Explore



Unrolling, blocking and Mflops/s for matrix multiplication
Seymour K, You H, Dongarra J. A comparison of search heuristics for empirical code optimization. InCLUSTER 2008 Oct 1 (pp. 421-429)

- Represent the effect of all possible configurations on the objectives, can be difficult to explore, with multiple local optima and undefined regions
- Main issues are exponential growth, geometry, & measurement time

## A Transparent Design of Experiments Approach



- An initial model is provided by the user (steps 1 & 2)
- Design of Experiments (DoE) guides exploration (steps 3 & 4)
- Significant factors are identified by Analysis of Variance (ANOVA) (steps 5 & 6)
- New fitted model predicts best value for significant factors (steps 7 & 8)

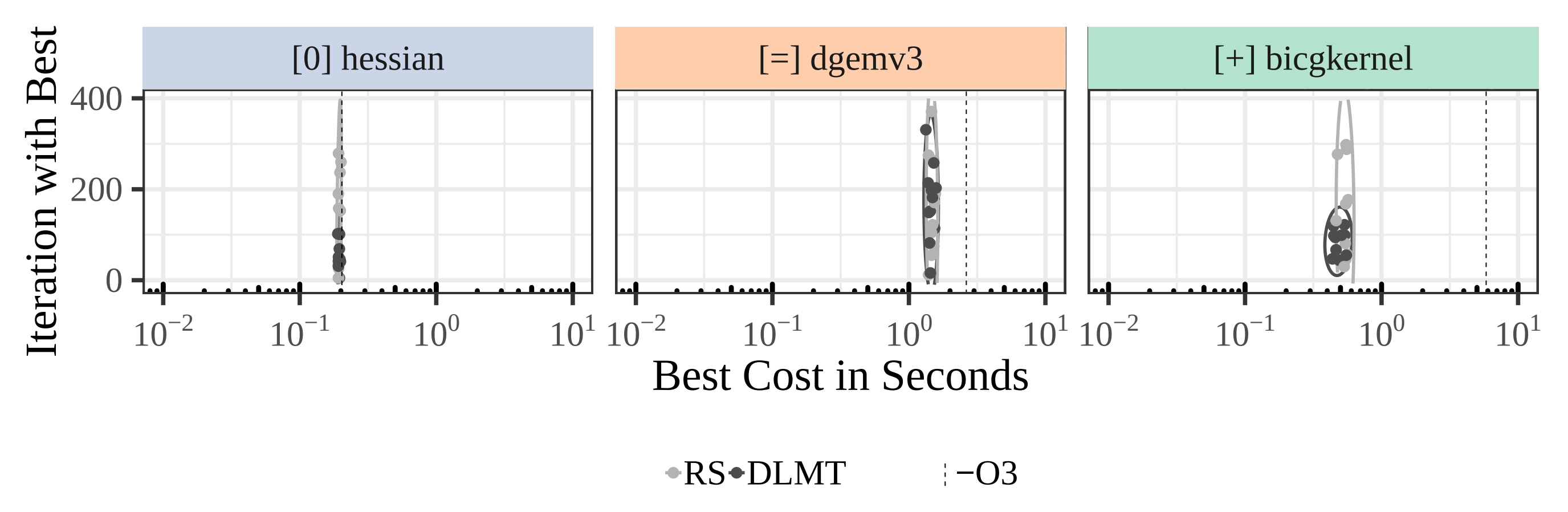
**Transparent:** factor and level selections based on ANOVA

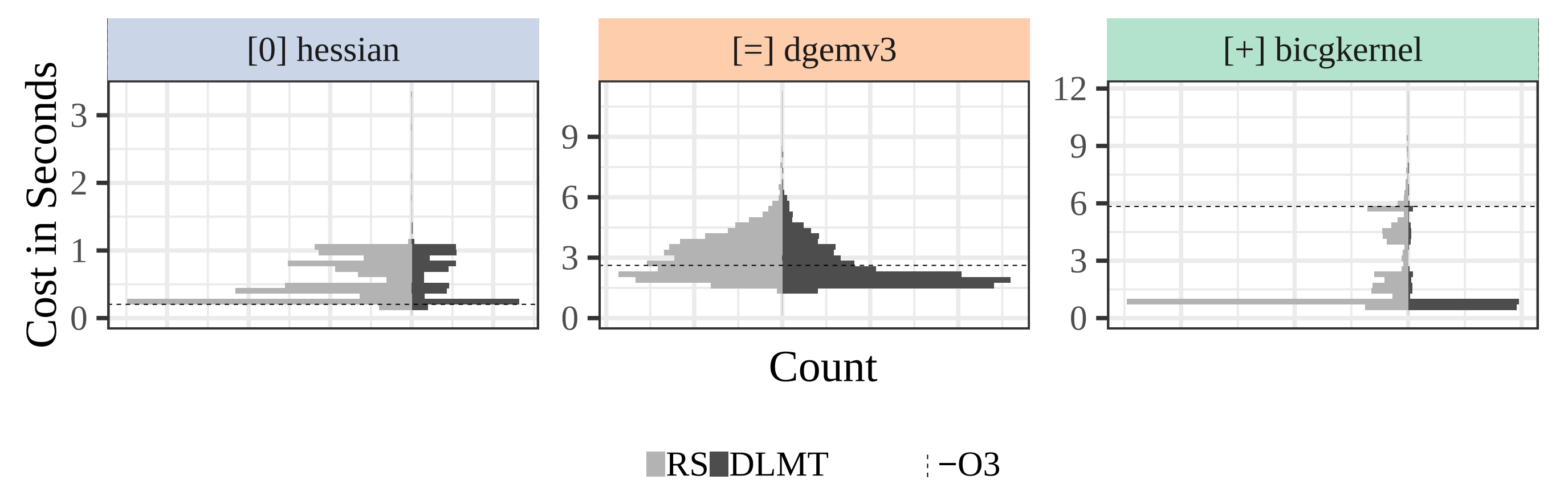
**Parsimonious:** DoE decreases measurements

## Extensive Evaluation on the SPAPT Benchmark

- SPAPT is an autotuning benchmark for CPU kernels, with search space sizes between $10^7$ and $10^{36}$
- We evaluated DLMT on 11 kernels (3 shown below) using the same initial performance model, and fixed budget

Our approach (DLMT) achieved good speedups using a smaller budget, while exploring better configurations





Laboratoire d'informatique de Grenoble 2019