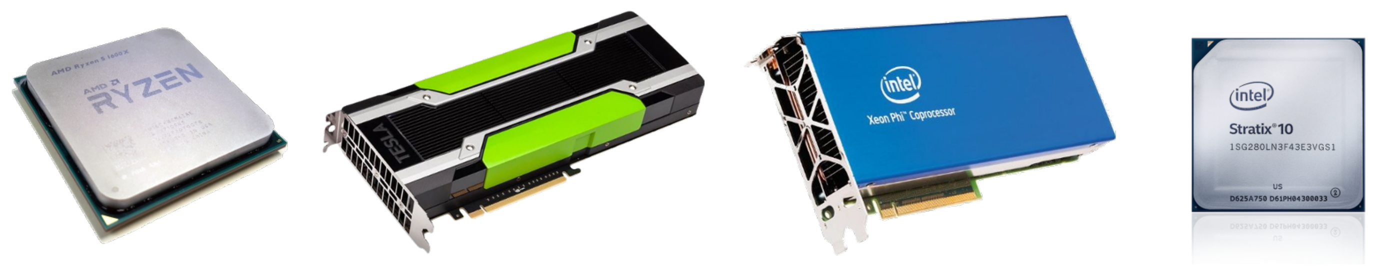
Autotuning under Tight Budget Constraints: A Transparent Design of Experiments Approach

Pedro Bruel[1,3], *Steven Quinito Masnada*[2], *Brice Videau*[3], *Arnaud Legrand*[3], *Jean-Marc Vincent*[3], *Alfredo Goldman*[1]

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

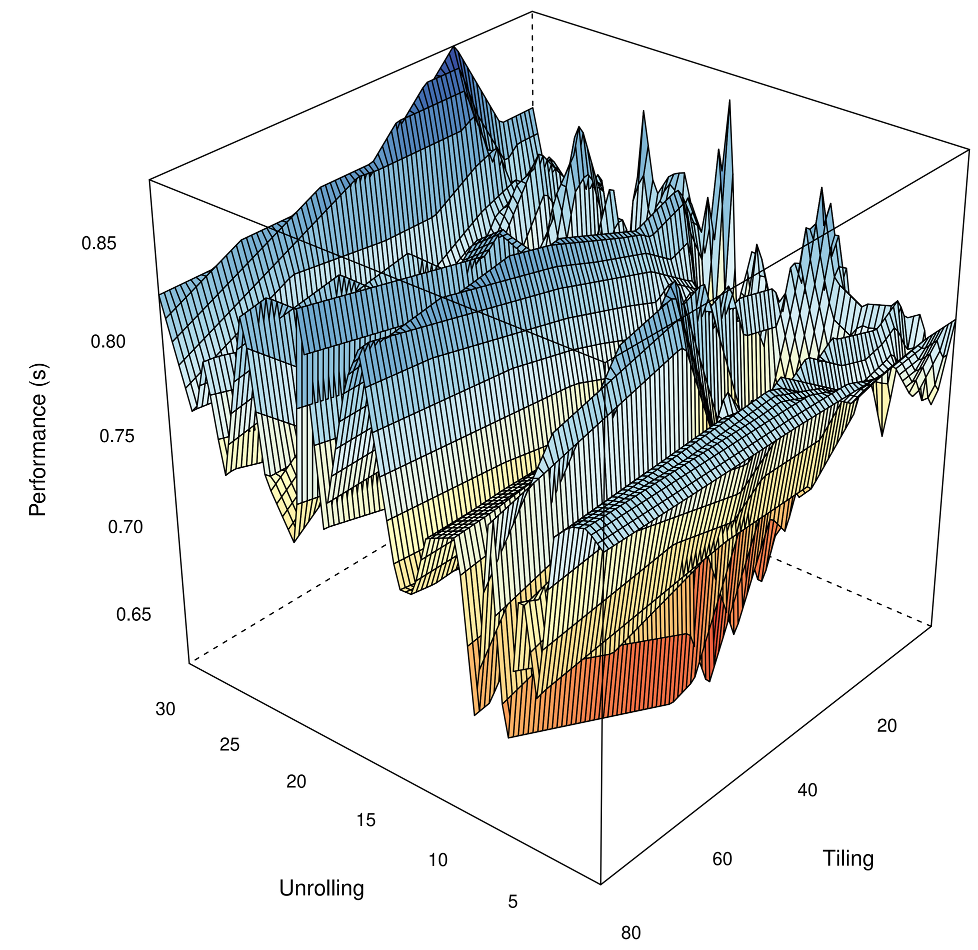
System	Domain	Approach
ATLAS	Dense Linear Algebra	Exhaustive
INSIEME	Compiler	Genetic Algorithm
Active Harmony	Runtime	Nelder-Mead
ParamILS	Domain-Agnostic	Stochastic Local Search
OPAL	Domain-Agnostic	Direct Search
OpenTuner	Domain-Agnostic	Ensemble
MILEPOST GCC	Compiler	Machine Learning
Apollo	GPU kernels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

Exhaustive, Meta-Heuristics, Machine Learning

Assumptions:

- Many measurements, "smoothness", reachable solutions After optimizing:
  - Learn "nothing", can't explain choices

Autotuning: Search Spaces are Hard to Explore



Unrolling, tiling and performance for a biconjugate gradient kernel

- 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

Design of Experiments: Exploration under a Budget

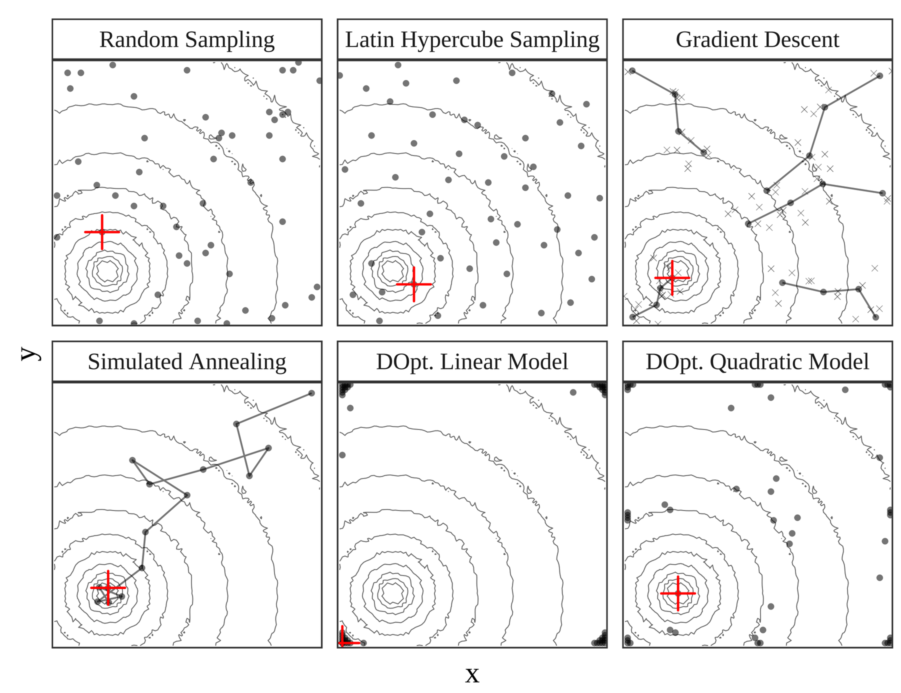
Design of Experiments (DoE):

- Factors are program parameters, and levels are possible factor values
- An experiment fixes levels, and a design is a selection of experiments to run
- A performance model is required to construct designs

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

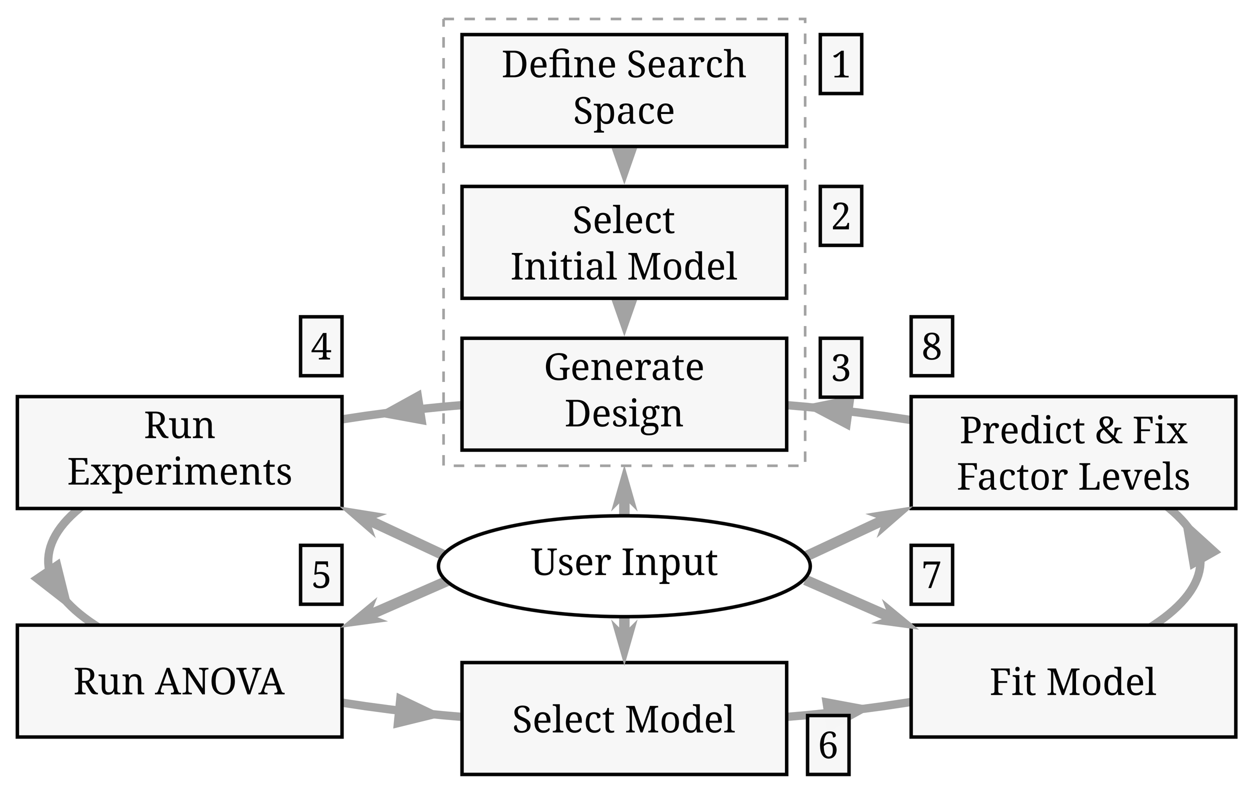
A Plackett-Burman design for 7 2-level factors

- Results, or responses, 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 Transparent Design of Experiments Approach



- An initial model is provided by the user (steps 1 & 2)
- Design of Experiments 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

A Motivating Result on a GPU Kernel

- Kernel factors:

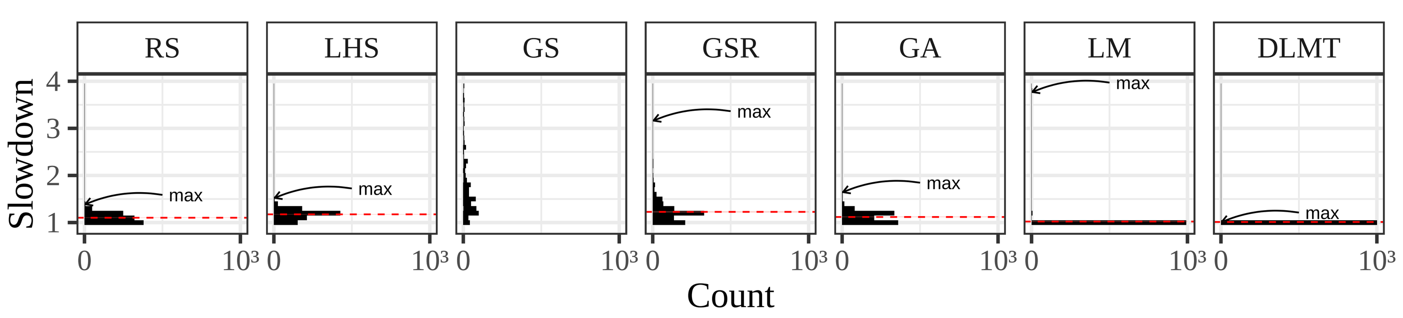
Factor	Levels	Short Description
vector_length	$2^0, \dots, 2^4$	Size of support arrays
load_overlap	*true*, *false*	Load overlaps in vectorization
temporary_size	$2, 4$	Byte size of temporary data
elements_number	$1, \dots, 24$	Size of equal data splits
y_component_number	$1, \dots, 6$	Loop tile size
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

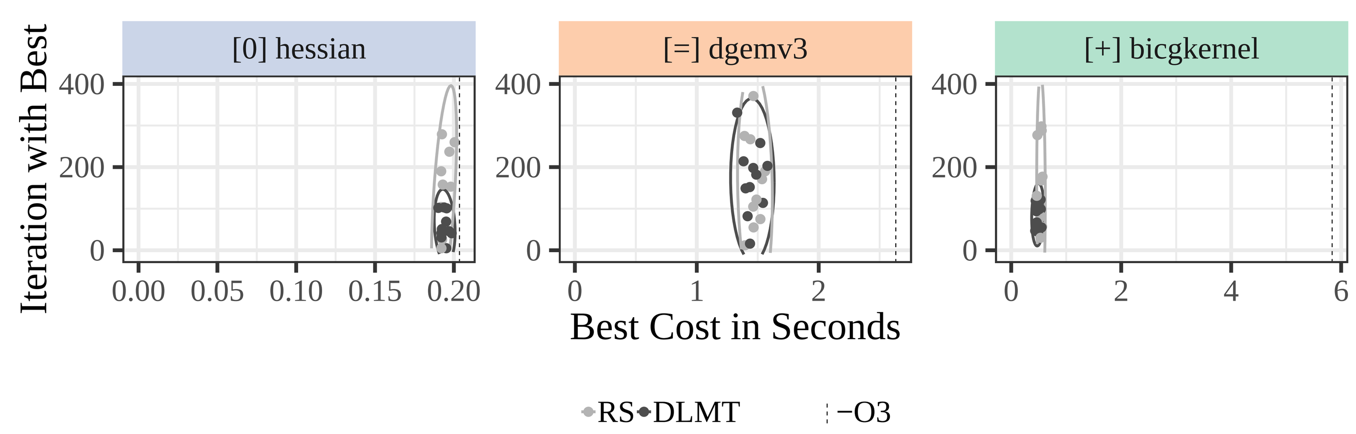


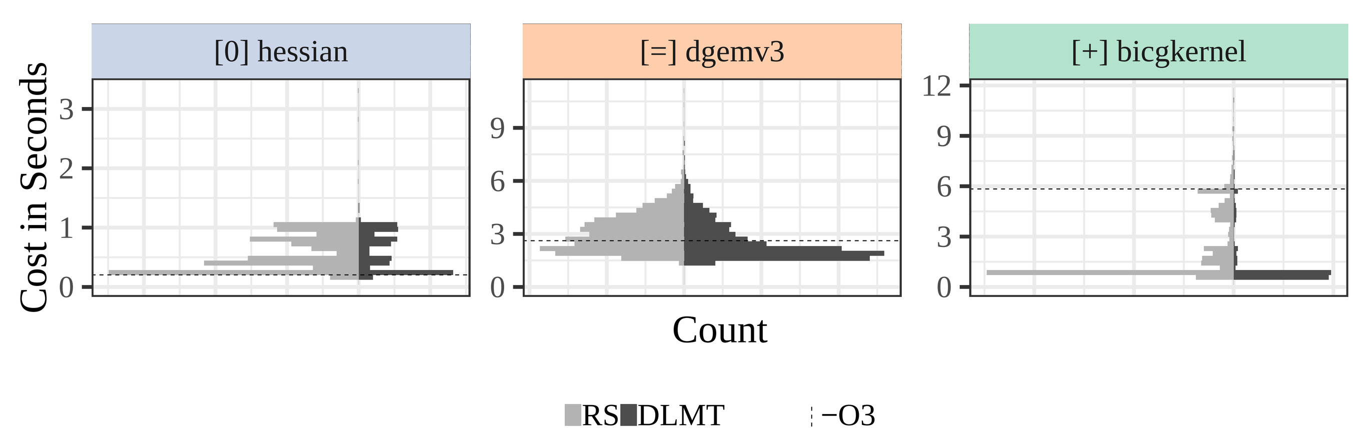
RS	LHS	GS	GSR	GA	LM	DLMT
Random Sampling	Latin Hyper Square	Greedy Search	Greedy with Restart	Generic Algorithm	Linear Model	Our DoE Approach

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 17 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

[1]University of São Paulo, São Paulo, Brazil, with CAPES Funding
[2]University of Grenoble Alpes, Inria, CNRS, Grenoble INP, LJK 38000 Grenoble, France
[3]University of Grenoble Alpes, CNRS, Inria, Grenoble INP, LIG 38000 Grenoble, France

POLARIS - Systèmes Répartis, Calcul Parallèle et Réseaux

L I G