# A Design of Experiments Approach to Autotuning under Tight Budget Constraints

Pedro Bruel*†, Arnaud Legrand*, Brice Videau* and Alfredo Goldman†
*University of Grenoble Alpes, CNRS, INRIA, LIG - Grenoble, France
Email: {arnaud.legrand, brice.videau}@imag.fr
†University of São Paulo - São Paulo, Brazil
Email: {phrb, gold}@ime.usp.br

***Abstract*—Abstract**

## I. Introduction

Optimizing code for objectives such as performance and power consumption is fundamental to the success and cost effectiveness of industrial and scientific endeavors in High Performance Computing. A considerable amount of highly specialized time and effort is spent in porting and optimizing code for GPUs, FPGAs and other hardware accelerators. Experts are also needed to leverage bleeding edge software improvements in compilers, languages, libraries and frameworks. The automatic configuration and optimization of High Performance Computing applications, or *autotuning*, is a technique effective in decreasing the cost and time needed to adopt efficient hardware and software. Typical targets for autotuning include algorithm selection, source-to-source transformations and compiler configuration.

Autotuning can be studied as a search problem, where the objective is to minimize single or multiple software of hardware metrics. The exploration of the search spaces defined by configurations and optimizations present interesting challenges to search strategies. These search spaces grow exponentially with the number of considered configuration parameters and their possible values. They are also difficult to extensively explore due to the often prohibitive costs of hardware utilization and program compilation and execution times. Developing autotuning strategies capable of producing good optimizations while minimizing resource utilization is therefore essential. The capability of acquiring knowledge about an optimization problem is also a desired feature of an autotuning strategy, since this knowledge can decrease the cost of subsequent optimizations of the same application or for the same hardware.

It is common and usually effective to use search meta-heuristics such as genetic algorithms and simulated annealing in autotuning. These strategies usually attempt to exploit local properties and are not capable of fully exploiting global search space structures. They are also not much more effective in comparison with a naive uniform random sample of the search space [1], [2], and usually rely on a large number of measurements and frequent restarts to achieve good performance improvements. Search strategies based on gradient descent also are commonly used in autotuning and rely on a large number of measurements. Their effectiveness diminishes additionally in search spaces with complex local structures. Completely automated machine learning autotuning strategies are effective in building models for predicting important optimization parameters, but still rely on a sizable data set for training. Large data sets are fundamental to strategies based on machine learning since they select models from a generally very large class.

Search strategies based on meta-heuristics, gradient descent and machine learning require a large number of measurements to be effective, and are usually incapable of providing knowledge about search spaces to users. At the end of each autotuning session it is difficult to decide if and where further exploration is warranted, and impossible to know which parameters are responsible for the observed improvements. After exploring a search space, it is impossible to confidently deduce its global properties since its was explored with unknown biases.

In this paper we propose an autotuning strategy that leverages existing expert and approximate knowledge about a problem in the form of a performance model, and refines this initial model iteratively using empirical performance evaluations, statistical analysis and user input. Our strategy puts a heavy weight on decreasing the costs of autotuning by using efficient Design of Experiments strategies to minimize the number of experiments needed to find good optimizations. Each optimization iteration uses *Analysis of Variance* (ANOVA) to help identify the relative significance of each configurable parameter to the performance observations. An architecture- and problem-specific performance model is built iteratively and with user input, enabling informed decisions on which regions of the search space are worth exploring.

We present the performance of our approach on a Laplacian Kernel for GPUs where the search space, global optimum and performance model approximation are known. The experimental budget on this application were tightly constrained. The speedups achieved and the budget utilization of our approach on this setting motivated a more comprehensive performance evaluation. We chose the *Search Problems in Automatic Performance Tuning* (SPAPT) [3] benchmark for this evaluation, where our approach was able to find speedups of over $50\times$ for some SPAPT applications, finding speedups better than random sampling in some scenarios. Despite using

generic performance models for every SPAPT application, our approach was able to significantly decrease the budget used to find performance improvements.

The rest of this paper is organized as follows. Section II presents related work on source-to-source transformation, which is the main optimization target in SPAPT problems, on autotuning systems and on search space exploration strategies. Section III presents a detailed description of the implementation of our approach and its background. It discusses the Design of Experiments concepts we incorporate, and the ANOVA and linear regression algorithms we use in analysis steps. Section IV presents our results with the GPU Laplacian Kernel and the SPAPT benchmark. Section V discusses our conclusions and future work.

## II. Background

### A. Source-to-source Transformation

### B. Autotuning

### C. Search Space Exploration Strategies

## III. Applying Design of Experiments to Autotuning

### A. Design of Experiments

#### 1) D-Optimal Designs:

### B. ANOVA

## IV. Performance Evaluations

### A. Example on a GPU Laplacian Kernel

### B. Results on the SPAPT Benchmark

#### 1) Experimental Methodology:

## V. Conclusion

## Acknowledgment

## References


[1] K. Seymour, H. You, and J. Dongarra, "A comparison of search heuristics for empirical code optimization." in *CLUSTER*, 2008, pp. 421–429.

[2] P. M. Knijnenburg, T. Kisuki, and M. F. O'Boyle, "Combined selection of tile sizes and unroll factors using iterative compilation," *The Journal of Supercomputing*, vol. 24, no. 1, pp. 43–67, 2003.

[3] P. Balaprakash, S. M. Wild, and B. Norris, "Spapt: Search problems in automatic performance tuning," *Procedia Computer Science*, vol. 9, pp. 1959–1968, 2012.