Tuning the tuner: the art of automatic performance optimization

Willemsen, F.Q.

1 Introduction

Modern High-Performance Computing (HPC) systems increasingly rely on heterogeneous architectures, combining multicore CPUs with accelerators such as Graphics Processing Units (GPUs). These platforms enable major advancements in fields including artificial intelligence, climate modeling, and radio astronomy, but achieving high performance remains challenging. Application performance depends on numerous interdependent implementation choices, such as parallelization strategies, data layouts, and algorithmic variants, whose effects on performance are often non-linear, hardware-dependent, and hard to predict. The combinatorial explosion of possible parameter configurations, coupled with hardware and software constraints, makes manual tuning infeasible.

Auto-tuning addresses this challenge by automating the search for high-performing configurations. Auto-tuners systematically generate and evaluate program variants, varying performance-critical parameters to optimize a chosen metric such as execution time or energy efficiency. While effective, auto-tuning introduces a new optimization problem: how to design and configure the auto-tuning process itself. Choices such as search space construction, optimization algorithm evaluation and selection, and constraint handling have a substantial impact on tuning efficiency and results.

This thesis investigates the *optimization of auto-tuning problems* with a focus on GPU kernel optimization and HPC workloads. It develops methods to make auto-tuning more efficient, scalable, and robust.

Thesis Statement: The central aim of this thesis is to advance the efficiency, scalability, and robustness of auto-tuning by establishing methods for evaluating and improving optimization algorithms, search space construction, hyperparameter tuning, and constraint handling in the context of GPU and HPC applications.

1

1.1 Auto-tuning

High-Performance Computing (HPC) has become a cornerstone for advancing scientific discovery and industrial innovation, powering applications in domains such as climate modeling, artificial intelligence, quantum physics, fluid dynamics, and radio astronomy [66, 87, 123, 110]. The massive parallelism of GPUs and the increasing integration and reliance of modern HPC systems on GPUs has enabled breakthroughs across disciplines. However, fully exploiting the computational potential of these platforms remains challenging [167, 70].

Programming and optimizing for GPUs and other accelerators involves making numerous interdependent design decisions that have a complex and often unpredictable impact on performance. These decisions include mapping computations to parallel threads and thread blocks, selecting algorithms and data structures, organizing data in different memory hierarchies, and applying code transformations such as loop tiling, vectorization, or unrolling. Many of these choices are governed by discrete parameters with a wide range of valid values, such as block sizes, tile dimensions, and data layouts. Moreover, a large fraction of the possible parameter combinations can be infeasible due to hardware limits, e.g., exceeding shared memory capacity, register file size, or other architectural constraints, or implementation-specific limitations.

The performance optimization process is therefore inherently combinatorial: the set of all possible parameter combinations forms a vast, discrete search space. Only a small fraction of configurations typically yield near-optimal performance [136, 134, 165]. Manually exploring these spaces is infeasible due to their size, the complexity of hardware–software interactions, and the cost of compiling and executing each candidate configuration.

Auto-tuning addresses this challenge by automating the search for high-performing configurations [9]. Auto-tuning frameworks systematically generate and evaluate program variants by varying tunable parameters; performance-critical parameters that govern aspects such as thread geometry, work distribution, memory layout, and algorithmic strategies. Evaluation typically involves compiling the variant (if necessary), executing it on the target hardware, measuring relevant metrics such as execution time or energy consumption, and validating correctness.

Over the years, auto-tuning has been successfully applied in a variety of domains, yielding high-performance libraries such as FFTW for Fast Fourier Transforms [50] and ATLAS for linear algebra [168], as well as specialized frameworks for domains including image processing [101, 163], fluid dynamics [44], and radio astronomy [137]. General-purpose auto-tuning frameworks [3, 114, 125, 165] provide application-independent mech-

anisms for defining tunable parameters and systematically exploring the resulting search spaces.

Since exhaustive exploration is rarely feasible, auto-tuners often employ optimization algorithms to guide the search efficiently. These algorithms aim to identify high-performing configurations within acceptable time budgets, balancing exploration of the search space with exploitation of promising regions. Depending on the application, the optimization objective may be to maximize performance, minimize energy consumption, or optimize a combination of metrics.

In short, auto-tuning represents a critical intersection of performance engineering, empirical measurement, and combinatorial optimization. It provides a systematic way to bridge the gap between the theoretical capabilities of modern computing hardware and the practical performance achieved by applications, making it a fundamental tool in the HPC optimization toolkit.

1.2 Research Questions

While auto-tuning has emerged as a powerful method to automate performance optimization, several fundamental challenges remain unresolved. First, the search spaces defined by tunable parameters are vast, highly discrete, and often constrained, making exhaustive exploration infeasible. Many parameter configurations are invalid due to hardware or implementation limitations, and current search strategies struggle to balance efficiency with robustness in such irregular domains.

Second, the optimization algorithms that drive auto-tuning are difficult to evaluate and compare in a reproducible manner. Studies often report results under differing conditions, budgets, and metrics, preventing meaningful comparisons between methods. Without a common methodology, the community lacks a systematic way to assess algorithmic advances.

Third, constructing search spaces themselves can become a scalability bottleneck. For applications with large search space and complex constraints, resolving the search space may take minutes to days, limiting the practicality of auto-tuning at larger problem scales.

Fourth, while optimization algorithms have hyperparameters that strongly influence their behavior, these hyperparameters are rarely tuned in practice. This neglect leaves significant performance improvements untapped and obscures our understanding of how to configure optimizers effectively.

Fifth, existing optimization methods often waste effort evaluating infeasible configurations. Although constraint-aware techniques are common in related fields, they have yet to be systematically developed and applied in the context of auto-tuning.

1

Finally, recent advances in Large Language Models (LLMs) raise the possibility of automatically generating bespoke optimization algorithms, tailored to the characteristics of individual search spaces and applications. However, it is unclear whether LLMs can produce optimizers that are both effective and robust in an auto-tuning setting, and whether incorporating knowledge of the search space into their design process can provide a tangible advantage over human-crafted approaches.

The work presented in this thesis is guided by the following question:
How can we design, evaluate, and enhance optimization methods to make auto-tuning GPU kernels more effective, efficient, and adaptable across diverse problem settings? This is, in turn, addressed in the following research questions:

RQ1 (Chapter 2) Can Bayesian optimization be applied for more effective performance tuning of GPU kernels?

RQ2 (Chapter 3) How can optimization algorithms for auto-tuning be compared in a fair and reproducible manner?

RQ3 (Chapter 4) How can large search spaces for auto-tuning be constructed efficiently while minimizing infeasible configurations?

RQ4 (Chapter 5) Can hyperparameter optimization be used to improve the effectiveness of auto-tuners?

RQ5 (Chapter 6) Can constrained optimization techniques be applied to handle infeasible configurations in auto-tuning problems?

RQ6 (Chapter 7) Can Large Language Models, when guided by problem-specific knowledge, generate effective bespoke optimization algorithms for auto-tuning?

1.3 Main Contributions of this Thesis

[170] F.-J. Willemsen, R. van Nieuwpoort, and B. van Werkhoven. "Bayesian Optimization for auto-tuning GPU kernels". In: *2021 International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS) at SuperComputing 2021*. 2021 International Workshop on Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS) at SuperComputing 2021. Nov. 2021, pp. 106–117.

[173] F.-J. Willemsen, R. Schoonhoven, J. Filipovič, J. O. Tørring, R. van Nieuwpoort, and B. van Werkhoven. "A methodology for comparing optimization algorithms for auto-tuning". In: *Future Generation Computer Systems* 159 (2024), pp. 489–504.

[159] J. O. Tørring, B. van Werkhoven, F. Petrovč, F.-J. Willemsen, J. Filipovič, and A. C. Elster. "Towards a benchmarking suite for kernel tuners". In: *2023 IEEE international parallel and distributed processing symposium workshops (IPDPSW)*. IEEE, 2023, pp. 724–733.

[171] F.-J. Willemsen, R. V. van Nieuwpoort, and B. van Werkhoven. "Efficient construction of large search spaces for auto-tuning". In: *Proceedings of the 54th international conference on parallel processing (ICPP '25)*. San Diego, CA, USA and New York, NY, USA: Association for Computing Machinery, Sept. 2025 *(Best Paper Award)*.

[172] F.-J. Willemsen, R. V. van Nieuwpoort, and B. van Werkhoven. "Tuning the Tuner: Introducing Hyperparameter Optimization for Auto-Tuning". In: *Proceedings of the 2025 IEEE eScience conference*. 21st IEEE International Conference on eScience. Chicago, IL, USA, Sept. 2025.

[169] F.-J. Willemsen, S. Heldens, R. V. van Nieuwpoort, and B. van Werkhoven. "Constraint-aware Optimization in Auto-Tuning". In: *2026 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. New Orleans, LA, USA, 2026.

[174] F.-J. Willemsen, N. van Stein, and B. van Werkhoven. "Automated Algorithm Design for Auto-Tuning Optimizers". In: *Proceedings of Machine Learning and Systems*. MLSys 2026. Seattle, WA, USA, 2026.