Tuning the tuner: the art of automatic performance optimization

Willemsen, F.Q.

8 Conclusions and Outlook

This thesis has explored how auto-tuning can be made more efficient, scalable, and robust for heterogeneous High-Performance Computing (HPC), with a particular focus on GPU kernel tuning. By addressing challenges in search strategies, evaluation methodology, search space construction, hyperparameter tuning, constraint handling, and automated algorithm design, the work advances the state of the art in performance optimization for modern architectures through six distinct contributions.

Chapter 2 introduced a Bayesian Optimization framework adapted to the discrete and constrained nature of GPU auto-tuning. Through domain-specific design choices such as tailored kernel functions, robust exploration factors, and scalable acquisition functions, the framework consistently outperformed existing strategies. With this, **RQ1** on the suitability of Bayesian Optimization for GPU auto-tuning was answered positively, demonstrating that carefully designed, problem-aware methods outperform generic approaches.

Chapter 3 established a statistically sound methodology for evaluating optimization algorithms. By grounding comparisons in actual time spent and providing an implementation-independent baseline, the framework enables fair and reproducible evaluation. The accompanying open-source package encourages adoption by the community and beyond GPU auto-tuning, for instance, in broader algorithm configuration research. Thus, **RQ2** on how to fairly compare optimization algorithms was addressed, showing that reproducibility and methodological rigor are essential for meaningful progress in the field.

Chapter 4 developed a high-performance Constraint Satisfaction Problem solver to construct large, structured search spaces. Capable of handling billions of configurations in sub-second time, it removes a major bottleneck in auto-tuning workflows. Its integration into open-source tools ensures practical impact. This provides a clear answer to **RQ3** on how to efficiently construct and explore large, constrained search spaces, showing that search space generation no longer needs to be a limiting factor.

Chapter 5 focused on hyperparameter optimization for auto-tuners themselves. By enabling cost-effective large-scale experimentation in simulation mode and creating a FAIR

dataset, performance improvements exceeding 200% are achieved while substantially reducing the energy consumption and resource demand associated with hyperparameter tuning. These results underline the importance of systematic tuning not just for applications, but also for the optimizers that drive auto-tuning. This answers **RQ4**, showing that hyperparameter tuning of the tuner itself can yield substantial improvements in both effectiveness and sustainability.

Chapter 6 extended four evolutionary algorithms with constraint-awareness, improving their performance on the heavily constrained search spaces typical in GPU tuning. These results emphasize that constraints should be treated as first-class citizens in optimization. Consequently, **RQ5** on the impact of constraint handling in evolutionary algorithms was answered affirmatively, as constraint-awareness was shown to be crucial for efficiency and solution quality.

Chapter 7 investigated the automatic generation of optimization algorithms using Large Language Models. By embedding LLMs in a feedback loop with Kernel Tuner, it was shown that bespoke, problem-specific algorithms can outperform even hyperparameter-tuned human-designed methods by wide margins. Although still in an exploratory stage, this line of research suggests a path toward self-improving optimizers capable of adjusting to any problem provided by the user. Hence, **RQ6** was answered by demonstrating that automated algorithm design using LLMs is not only feasible but already competitive, with the potential to fundamentally reshape how optimizers are developed.

Taken together, the thesis demonstrates that progress in auto-tuning is not just made by exploring search spaces more exhaustively, but by improving the tuner itself. By making optimization algorithms domain-aware, rigorously evaluated, efficiently parameterized, constraint-conscious, and even automatically designed, substantial performance and usability gains are achieved, and in doing so, the work answers the overarching research question. The contributions show that careful design, such as by using methods with a robust mathematical foundation, fast and well-defined constraint handling, and hyperparameter tuning, produces optimizers that are tailored to the challenges of GPU auto-tuning. Robust evaluation methodologies ensure that progress is measurable, comparable, and reproducible across diverse problem settings. Continuous enhancement of optimizers, whether by fine-tuning their parameters or by generating entirely new algorithms with LLMs, demonstrates that auto-tuning systems can evolve alongside the hardware and applications they serve.

All methods and tools developed in this thesis are released as open-source software, ensuring reproducibility and enabling continued progress in the field. The advances described in this work enable entirely new problem scales and types to be tackled, not only

in auto-tuning research, but also in HPC applications at large. This includes domains such as climate modeling, astronomy, chemistry, and artificial intelligence, where performance gains and reduced energy use directly benefit both science and society. As heterogeneous HPC systems become ever more central to scientific and industrial computing, the approaches developed here lay the foundations for auto-tuning that is more adaptive, transparent, sustainable, and scalable.

Looking forward, the findings open several promising directions. The methodology, standards, and resources provided in this thesis are already being adopted by the auto-tuning community, providing a shared basis of analysis that enables comparison and reproducibility, thereby fostering scientific growth. The seeding of optimization algorithms and initial sampling methods with prior knowledge of auto-tuning problems using transfer learning could further accelerate the tuning process. Multi-objective optimization and optimization toward energy-performance trade-offs offer natural next steps with substantial practical impact. Finally, automated algorithm design, while still in its infancy, suggests the possibility of self-adapting optimizers that evolve alongside the systems and applications they target. Together, these directions point toward a future where auto-tuning not only meets the growing demands of high-performance computing but also contributes to more sustainable and impactful scientific and industrial computing.