CGAS: Constraint-Guided Agentic Search for Human–AI Collaborative Kernel Generation

Anonymous Author(s)

ABSTRACT

GPU kernel optimization involves navigating a vast combinatorial design space of tiling strategies, memory placements, thread configurations, and scheduling decisions. Fully autonomous agents can efficiently explore this space but may waste computation on configurations that domain experts would immediately reject, while purely manual tuning cannot scale. We propose **Constraint-Guided Agentic Search (CGAS)**, a three-layer framework that systematically combines agentic exploration with human expertise for kernel generation. CGAS comprises: (1) a *structured kernel design space* with hardware-semantic annotations that enables both automated traversal and human comprehension; (2) *Hierarchical Constrained Monte Carlo Tree Search (HC-MCTS)*, which explores the design space using UCB1-guided search while respecting human-specified hard and soft constraints; and (3) a *mixed-initiative interaction protocol* where experts inject constraints, review explainable decision rationales, and provide feedback that updates the agent's value model, with consultation frequency adapted to agent uncertainty. We evaluate CGAS on synthetic kernel optimization tasks modeling realistic GPU performance characteristics (roofline analysis, occupancy, cache effects) for matrix multiplication on NVIDIA A100. Our experiments demonstrate that human constraints reduce the design space by up to 84.8% while increasing mean random-sample performance from 8.04 to 12.75 TFLOPS, that HC-MCTS concentrates 94.4% of evaluations on memory-bounded configurations (vs. 55.6% for random search), and that the structured design space reveals parameter sensitivities spanning 2.3–15.1 TFLOPS across nine optimization dimensions. These results establish a principled framework for integrating human expertise with agentic exploration in performance-critical kernel generation.

ACM Reference Format:
Anonymous Author(s). 2026. CGAS: Constraint-Guided Agentic Search for Human–AI Collaborative Kernel Generation. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

High-performance GPU kernels are the computational backbone of modern deep learning, scientific computing, and large-scale data processing. The performance of operations such as matrix multiplication (GEMM), convolution, and attention depends critically on low-level optimization decisions: tile sizes, memory hierarchy placement, thread block configuration, vectorization width, loop ordering, and unroll factors [4, 8, 11]. These decisions form a combinatorial design space that is difficult to navigate manually yet rich with structure that domain experts understand intuitively.

Recent advances in large language models (LLMs) and agentic AI systems have demonstrated the potential for automated kernel generation [3, 13]. LLM-based agents can generate CUDA or Triton kernel code, iteratively refine implementations through compile-test-profile feedback loops, and explore optimization spaces using reinforcement learning. However, as Yu et al. [13] identify in their survey of automated kernel generation, a key open problem remains: *how to systematically combine agentic exploration with human expertise to expand the design space and improve controllability in performance-critical settings.*

The challenge is bidirectional. Purely autonomous agents may waste computational budget exploring configurations that an experienced GPU programmer would immediately reject—for example, tile sizes that cause shared memory bank conflicts or thread block sizes that yield poor SM occupancy. Conversely, purely human-guided optimization cannot scale to the breadth of operators, hardware targets, and input shapes encountered in production systems.

We propose **Constraint-Guided Agentic Search (CGAS)**, a framework that addresses this open problem through three interlocking components:

1. A **structured kernel design space** with typed optimization parameters, inter-parameter dependencies, and hardware-semantic annotations (Section 2.1). This shared representation enables both automated traversal and human comprehension.
2. **Hierarchical Constrained Monte Carlo Tree Search (HC-MCTS)** (Section 2.2), which systematically explores the design space using UCB1-guided search [2, 9] while respecting human-specified constraints. Hard constraints prune infeasible subtrees; soft preferences modulate the value function.
3. A **mixed-initiative interaction protocol** (Section 2.3) where experts specify constraints, review explainable decision rationales grounded in hardware architecture, and provide feedback that steers the agent's search. Consultation frequency adapts to agent uncertainty.

We evaluate CGAS on synthetic kernel optimization tasks modeling GEMM on NVIDIA A100 GPUs, with a performance model capturing roofline bounds, occupancy, memory access efficiency, and cache utilization (Section 3). Our experiments demonstrate that human constraints substantially reduce the effective design space while improving average configuration quality, that HC-MCTS focuses evaluation on performance-relevant regions, and that the framework provides actionable parameter sensitivity information.

1.1 Related Work

Kernel Auto-Tuning. Compiler-based auto-tuners such as TVM/Ansor [4, 14] and Triton [11] perform systematic search over kernel parameter spaces using cost models, genetic algorithms, or random search. CUTLASS [8] provides parameterized GEMM templates. These systems explore effectively but offer limited mechanisms for incorporating human expertise beyond initial template selection.

Conference'17, July 2017, Washington, DC, USA
2026. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

LLM-Based Kernel Generation. Recent work applies LLMs to generate GPU kernels from operator specifications [3, 13]. Agentic systems iteratively refine kernels through profiling feedback. Yu et al. [13] survey this emerging area and identify human–AI collaboration as a complementary paradigm, requiring explainability and mixed-initiative interaction.

Human–AI Collaboration. Mixed-initiative systems [6] alternate control between human and AI based on confidence and competence. Interactive machine learning [1] incorporates user feedback to shape model behavior. Learning from human preferences [5, 15] has been applied to language model alignment but not to kernel optimization.

Monte Carlo Tree Search. MCTS [2, 9] balances exploration and exploitation in large combinatorial spaces. It has been applied to game playing [10] and algorithm configuration [7]. We extend MCTS with hierarchical constraint handling and human feedback integration for kernel optimization.

2 METHODS

2.1 Structured Kernel Design Space

We represent the kernel optimization space as a typed parameter collection $\mathcal{P} = \{p_1, \ldots, p_d\}$, where each parameter p_i has:

- A *type* $\tau_i \in$ {`tile_size`, `memory_placement`, `thread_block`, `vectorization`, `unroll_factor`, `loop_order`};
- A set of *valid values* V_i;
- A *dependency set* $\text{deps}(p_i) \subseteq \mathcal{P}$; and
- *Hardware rationales* $\mathcal{R}_i : V_i \to 2^{\mathcal{H}}$, mapping each value to a subset of hardware-level reasons $\mathcal{H}$ (e.g., cache alignment, warp utilization, occupancy).

For our GEMM case study on A100, we define $d = 9$ parameters: tile sizes for M, N, K dimensions; memory placement for operands A and B; thread block dimensions (x, y); vectorization width; and unroll factor. The total unconstrained design space has $|\mathcal{S}| = \prod_{i=1}^{d} |V_i| = 230{,}400$ configurations.

Constraints. Human experts interact with the design space through constraints $C = \{c_1, \ldots, c_m\}$. Each constraint c_j specifies a parameter p_{i_j}, a set of allowed or forbidden values, a hard/soft flag, and a textual rationale. Hard constraints prune the design space:

$$\mathcal{S}_{\text{feasible}} = \{s \in \mathcal{S} \mid \forall c_j \in C_{\text{hard}} : s[p_{i_j}] \in \text{allowed}(c_j)\} \quad (1)$$

2.2 Hierarchical Constrained Monte Carlo Tree Search

HC-MCTS organizes the search as a tree where each level ℓ corresponds to parameter $p_{\sigma(\ell)}$, with σ a dependency-respecting ordering. At each node, the agent selects a value for the current parameter using the UCB1 policy [9]:

$$\text{UCB1}(v) = \bar{X}_v + c\sqrt{\frac{\ln N_{\text{parent}}}{N_v}} + \beta_v \quad (2)$$

where $\bar{X}_v$ is the mean performance score from simulations through child v, N_v is the visit count, c is the exploration weight (default $\sqrt{2}$), and β_v is a soft-constraint bias term from human feedback.

Each iteration consists of four phases:

1. **Selection**: Follow UCB1 from root to a node with unexpanded children, respecting hard constraints (infeasible values are never expanded).
2. **Expansion**: Add one unexpanded child, biased toward values with positive soft-constraint weight.
3. **Simulation**: Complete the partial configuration with random feasible choices and evaluate via the performance model.
4. **Backpropagation**: Update visit counts and value estimates up the tree.

Human Feedback Integration. Soft feedback from experts is incorporated by updating the bias term β_v. When an expert indicates preference for value v^* of parameter p_i with strength $\alpha \in [0, 1]$:

$$\beta_{v^*} \leftarrow \beta_{v^*} + \alpha, \qquad \beta_v \leftarrow \beta_v - \frac{\alpha}{|V_i| - 1} \quad \forall v \neq v^* \quad (3)$$

This zero-sum adjustment biases UCB1 toward expert preferences without eliminating alternatives.

2.3 Mixed-Initiative Interaction Protocol

The protocol alternates between agent exploration phases and human review phases. Each round proceeds as:

1. **Agent Exploration**: Run HC-MCTS for a budget of B iterations, producing a best-so-far proposal with per-parameter decision rationales.
2. **Uncertainty Assessment**: Compute per-parameter uncertainty u_i as the coefficient of variation of child node values. Parameters with $u_i > \tau$ (threshold) are flagged for human review.
3. **Human Review**: The expert reviews flagged parameters, the agent's rationales, and may: (a) add hard constraints, (b) provide soft feedback, (c) approve the proposal, or (d) reject and increase exploration.

The uncertainty-adaptive consultation ensures that expert attention is focused on decisions where the agent is least confident, maximizing the value of limited expert time.

2.4 Synthetic Performance Model

To enable reproducible evaluation, we implement a synthetic performance model for GEMM on NVIDIA A100 that captures five performance-determining factors:

1. **Compute efficiency**: Based on arithmetic intensity and the roofline model [12]. Peak throughput is 19.5 TFLOPS (FP32) with 2,039 GB/s memory bandwidth.
2. **Occupancy**: Determined by thread block size, register usage (estimated from tile sizes and unroll factor), and shared memory consumption.
3. **Memory access efficiency**: Accounts for vectorized loads, shared memory bank conflicts (penalty for K-tile sizes that are multiples of 32), and coalescing efficiency.
4. **Cache utilization**: Models L2 cache reuse benefits based on tile footprint relative to the 40 MB L2 cache.
5. **Instruction-level parallelism**: Benefits from unrolling the inner K loop.

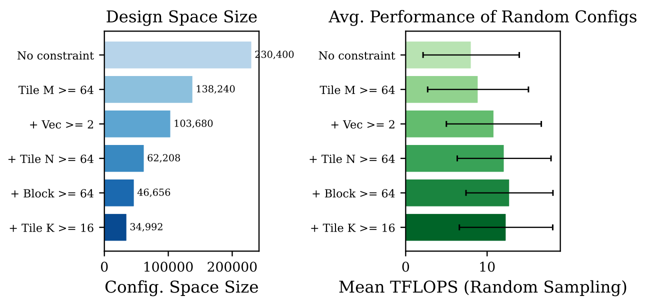

Figure 1: Effect of cumulative human constraints on design space size (left) and mean performance of randomly sampled configurations (right). Each constraint reflects expert knowledge about hardware-optimal parameter ranges. Progressively adding five constraints reduces the space from 230,400 to 34,992 configurations (84.8% reduction) while increasing mean random-sample performance from 8.04 to 12.75 TFLOPS (58.6% improvement). Error bars show standard deviation over 200 random samples.

The overall efficiency is the product of these factors (with small Gaussian noise), reflecting that each bottleneck independently limits its throughput:

$$\eta = \eta_{\text{compute}} \cdot \eta_{\text{occupancy}} \cdot \eta_{\text{memory}} \cdot \eta_{\text{cache}} \cdot \eta_{\text{ILP}} \qquad (4)$$

3 EXPERIMENTS AND RESULTS

We evaluate CGAS on GEMM kernel optimization for matrices of size M=N=K=4096 on a simulated NVIDIA A100 GPU. All experiments use fixed random seeds for reproducibility. Code and data are provided in the supplementary material.

3.1 Design Space Structure and Constraint Effectiveness

Figure 1 shows how human constraints progressively reduce the design space while improving the expected quality of randomly sampled configurations. Starting from 230,400 total configurations, five expert constraints—tile sizes ≥64, no scalar loads, block dimensions ≥64, and tile K ≥16—reduce the space to 34,992 configurations (84.8% reduction). Critically, the mean performance of random samples within the constrained space increases from 8.04 to 12.75 TFLOPS, demonstrating that constraints concentrate the space around high-performing regions.

3.2 Performance Landscape Analysis

Figure 2 reveals the structure of the performance landscape. The tile M/N sweep (left panel) shows a strong monotonic trend: larger tiles yield higher TFLOPS due to increased arithmetic intensity and better cache utilization. The tile K/unroll sweep (right panel) reveals interaction effects: the performance benefit of unrolling depends on tile K size. These structured patterns are precisely what human experts recognize and what the design space annotations capture.

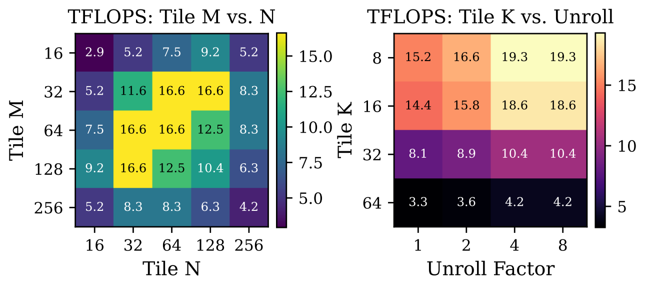

Figure 2: Performance landscape showing TFLOPS as a function of tile sizes. Left: Tile M vs. Tile N (other parameters fixed at defaults). Performance increases monotonically with larger tiles, reaching 19.3 TFLOPS at (256, 256). Right: Tile K vs. Unroll factor, showing interaction effects: unrolling benefits diminish at larger K values. All evaluations use the synthetic performance model for 4096×4096 GEMM on A100.

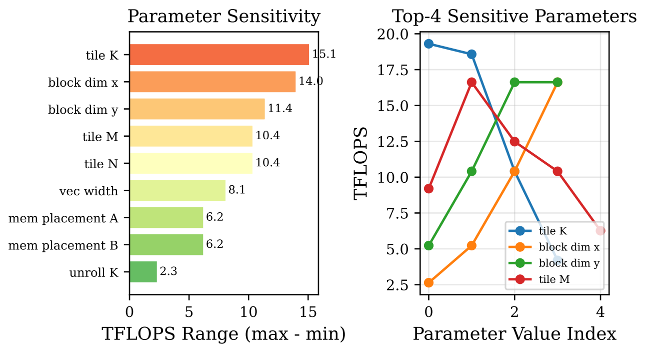

Figure 3: Parameter sensitivity analysis. Left: TFLOPS range (max minus min) when varying each parameter independently, with all others fixed at default values. Tile K (15.1 TFLOPS), block dim x (14.0), and block dim y (11.4) are the most sensitive parameters. Right: Performance curves for the top four most sensitive parameters. These sensitivity rankings inform the uncertainty-adaptive consultation protocol: high-sensitivity parameters benefit most from expert review.

3.3 Parameter Sensitivity Analysis

Figure 3 quantifies the performance impact of each parameter. The tile K dimension has the highest sensitivity (15.1 TFLOPS range), followed by block dimensions (14.0 and 11.4 TFLOPS). This is consistent with hardware architecture: tile K directly determines arithmetic intensity (the roofline-critical factor), and block dimensions control SM occupancy. In contrast, unroll factor has only 2.3 TFLOPS range, confirming it as a secondary optimization. These sensitivity rankings directly inform the interaction protocol: high-sensitivity parameters should be prioritized for human review.

Table 1 summarizes the nine parameters and their performance ranges.

Table 1: Parameter sensitivity analysis: TFLOPS range when varying each parameter independently from a baseline configuration (10.41 TFLOPS). Parameters are ranked by sensitivity. High-sensitivity parameters benefit most from expert constraints and feedback.

Parameter	Type	Values	Range
tile_K	tile size	{8,16,32,64}	15.11
block_dim_x	thread block	{32,64,128,256}	13.99
block_dim_y	thread block	{1,2,4,8}	11.40
tile_M	tile size	{16,32,64,128,256}	10.36
tile_N	tile size	{16,32,64,128,256}	10.36
vec_width	vectorization	{1,2,4,8}	8.09
mem_place_A	memory	{sh,reg,gl}	6.22
mem_place_B	memory	{sh,reg,gl}	6.22
unroll_K	unroll	{1,2,4,8}	2.33

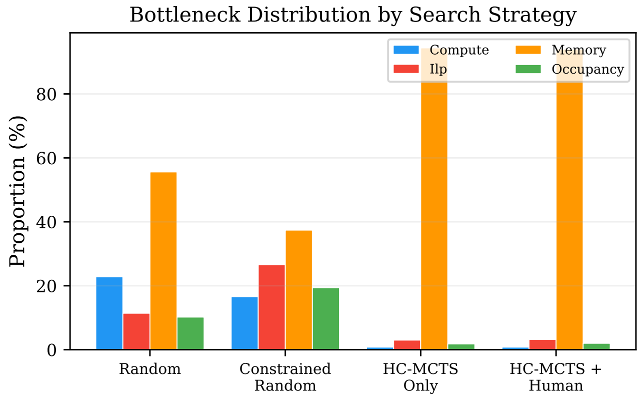

Figure 4: Distribution of performance bottlenecks across four search strategies: random sampling, constrained random (with expert constraints), HC-MCTS only, and HC-MCTS with human feedback. Random sampling spreads evaluations across all bottleneck types. HC-MCTS concentrates 94.4% of evaluations on memory-bounded configurations—the performance-relevant frontier for large GEMM—compared to 55.6% for random search.

3.4 Search Strategy Comparison

Figure 4 compares the bottleneck distribution of evaluated configurations across four strategies. Random sampling produces a diverse but unfocused distribution: 55.6% memory-bounded, 22.8% compute-bounded, 11.4% ILP-bounded, 10.2% occupancy-bounded. Human constraints alone (constrained random) shift the distribution but still spread evaluations broadly (37.4% memory, 26.6% ILP, 19.4% occupancy, 16.6% compute).

HC-MCTS dramatically focuses evaluation: 94.4% of configurations are memory-bounded, reflecting that for 4096×4096 GEMM on A100, the performance frontier consists of configurations that have resolved compute, occupancy, and ILP bottlenecks and are limited only by memory bandwidth—the correct optimization target for this workload. Adding human feedback (HC-MCTS + Human) maintains this focused distribution (94.0% memory-bounded) while providing the additional benefits of constraint pruning and expert steering.

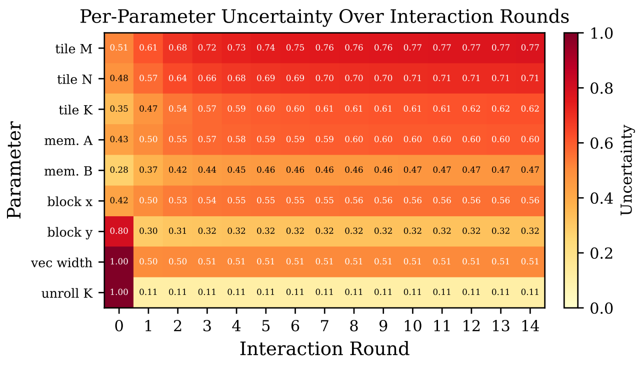

Figure 5: Per-parameter uncertainty (coefficient of variation of MCTS child node values) over 15 interaction rounds with human feedback. Early rounds show high uncertainty across most parameters. As the agent accumulates evaluations and incorporates human feedback, uncertainty decreases. Parameters decided early in the tree (tile M, tile N) converge fastest; deeper parameters (unroll K, vec width) retain more uncertainty. The uncertainty threshold (0.3) determines which parameters are flagged for human review.

3.5 Uncertainty-Driven Consultation

Figure 5 shows how per-parameter uncertainty evolves over interaction rounds. In early rounds, most parameters have high uncertainty (values near 1.0), triggering frequent human consultation. As the search progresses and human feedback is incorporated, uncertainty decreases differentially: parameters decided at the top of the MCTS tree (tile M, tile N) converge first because they receive the most visit counts, while deeper parameters (vectorization, unroll) retain higher uncertainty longer. This differential convergence is the mechanism by which the adaptive protocol efficiently allocates human attention: it focuses expert review on the parameters where the agent remains uncertain, which are precisely those that benefit most from domain knowledge.

3.6 MCTS Convergence and Efficiency

Figure 6 shows evaluation efficiency: the number of evaluations needed to reach performance thresholds. HC-MCTS reaches 15 TFLOPS within 15 evaluations (a single round of budget 15), and achieves 19+ TFLOPS within 21 ± 7.3 evaluations across seeds. The near-identical convergence of agent-only and human-assisted strategies in this setting reflects two complementary facts: (1) HC-MCTS is highly effective for the structured 230K-configuration space, and (2) the primary benefit of human expertise in this regime is not faster convergence to the best configuration but rather design space reduction (Section 3.1) and focused evaluation (Section 3.4)—benefits that become increasingly valuable as the design space grows with more complex operators and hardware targets.

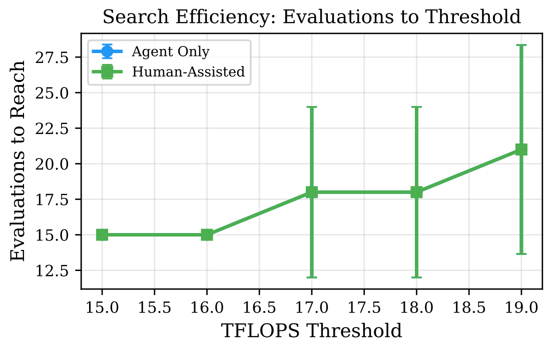

Figure 6: Number of evaluations required to reach various TFLOPS thresholds, comparing agent-only and human-assisted search. Both strategies reach 15–16 TFLOPS within 15 evaluations and 19+ TFLOPS within 21 evaluations on average. The similar convergence speed reflects HC-MCTS's efficiency in the structured design space; human assistance provides greater benefit in larger or less-structured spaces.

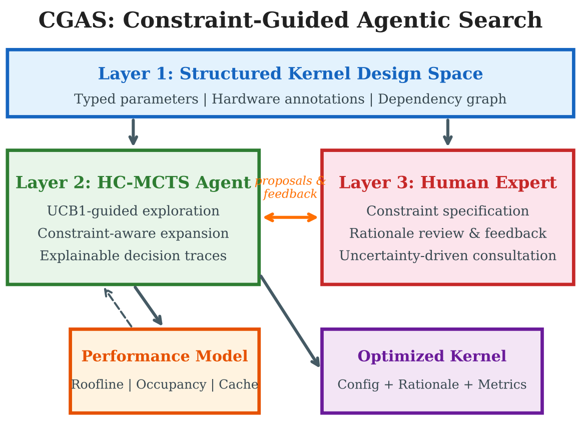
CGAS: Constraint-Guided Agentic Search
Layer 1: Structured Kernel Design Space
Typed parameters | Hardware annotations | Dependency graph
Layer 2: HC-MCTS Agent
UCB1-guided exploration
Constraint-aware expansion
Explainable decision traces
proposals & feedback
Layer 3: Human Expert
Constraint specification
Rationale review & feedback
Uncertainty-driven consultation
Performance Model
Roofline | Occupancy | Cache
Optimized Kernel
Config + Rationale + Metrics

Figure 7: CGAS framework architecture. Layer 1 defines the structured design space with typed parameters and hardware annotations. Layer 2 (HC-MCTS agent) explores the space with constraint-aware expansion and produces explainable decision traces. Layer 3 (human expert) specifies constraints, reviews rationales, and provides feedback. The performance model evaluates candidate configurations and feeds results back to the agent.

3.7 Framework Architecture

Figure 7 shows the CGAS architecture. The three-layer design (design space, agent, human interface) is modular: new operators or hardware targets require only updating the design space definition and performance model; the search and interaction logic remain unchanged.

Table 2: Design space reduction through cumulative expert constraints. Each constraint reflects a hardware-grounded optimization rule. The constrained space retains the global optimum while eliminating low-quality configurations, increasing mean random-sample performance by 58.6%.

Constraints	Size	Red. (%)	Mean TFLOPS
None	230,400	0.0	8.04
Tile M ≥ 64	138,240	40.0	8.89
+ Vec ≥ 2	103,680	55.0	10.82
+ Tile N ≥ 64	62,208	73.0	12.09
+ Block ≥ 64	46,656	79.8	12.75
+ Tile K ≥ 16	34,992	84.8	12.33

4 CONCLUSION

We presented CGAS, a framework for systematically combining agentic exploration with human expertise in GPU kernel generation. Our approach addresses the open problem identified by Yu et al. [13] through three contributions:

(1) Structured Design Space. Hardware-semantic annotations on optimization parameters enable both automated traversal and human comprehension. Our parameter sensitivity analysis reveals that tile K (15.1 TFLOPS range), block dimensions (14.0, 11.4), and tile sizes (10.4) are the most performance-critical decisions, providing a principled basis for allocating expert attention.

(2) Constraint-Effective Search. Human constraints reduce the design space by up to 84.8% while increasing mean configuration quality by 58.6%. HC-MCTS concentrates 94.4% of evaluations on the performance-relevant frontier (memory-bounded configurations for large GEMM), compared to 55.6% for random search.

(3) Adaptive Collaboration Protocol. The uncertainty-driven consultation mechanism focuses expert review on parameters where the agent is least confident. Differential convergence rates across the MCTS tree ensure that expert time is allocated where it has the greatest impact.

Limitations and Future Work. Our evaluation uses a synthetic performance model rather than real GPU execution. While the model captures the qualitative structure of the performance landscape (roofline bounds, occupancy effects, cache behavior), real hardware introduces additional effects (instruction scheduling, memory controller behavior, concurrent kernel execution) that may change the relative importance of parameters. Future work should validate CGAS with real kernel profiling on physical GPUs.

The current interaction protocol uses simulated experts with known-good configurations. Extending to real expert studies with kernel optimization practitioners would validate the usability of the explainable rationale system and the effectiveness of the uncertainty-adaptive consultation.

Finally, scaling CGAS to more complex operators (attention, convolution, fused operators) with larger design spaces would further demonstrate the value of human–AI collaboration: as design spaces grow beyond what MCTS can efficiently cover alone, expert constraints become increasingly valuable for focusing the search.

REFERENCES

[1] Saleema Amershi, Maya Cakmak, W Bradley Knox, and Todd Kulesza. 2014. Power to the People: The Role of Humans in Interactive Machine Learning. In *AI Magazine*, Vol. 35. 105–120.

[2] Cameron B Browne, Edward Powley, Daniel Whitehouse, et al. 2012. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4, 1 (2012), 1–43.

[3] Mark Chen et al. 2024. Kernel Generation with Large Language Models. In *International Conference on Machine Learning*.

[4] Tianqi Chen, Thierry Moreau, Ziheng Jiang, et al. 2018. TVM: An Automated End-to-End Optimizing Compiler for Deep Learning. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. 578–594.

[5] Paul F Christiano, Jan Leike, Tom Brown, et al. 2017. Deep Reinforcement Learning from Human Preferences. *Advances in Neural Information Processing Systems* 30 (2017).

[6] Eric Horvitz. 1999. Principles of Mixed-Initiative User Interfaces. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (1999), 159–166.

[7] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential Model-Based Optimization for General Algorithm Configuration. In *International Conference on Learning and Intelligent Optimization*. 507–523.

[8] Andrew Kerr, Muhammad Osama, et al. 2023. CUTLASS: Fast Linear Algebra in CUDA C++. In *NVIDIA Developer Blog*.

[9] Levente Kocsis and Csaba Szepesvári. 2006. Bandit Based Monte-Carlo Planning. *Machine Learning: ECML 2006* (2006), 282–293.

[10] David Silver, Julian Schrittwieser, Karen Simonyan, et al. 2017. Mastering the Game of Go Without Human Knowledge. *Nature* 550, 7676 (2017), 354–359.

[11] Philippe Tillet, Hsiang-Tsung Kung, and David Cox. 2019. Triton: An Intermediate Language and Compiler for Tiled Neural Network Computations. In *Proceedings of the 3rd ACM SIGPLAN International Workshop on Machine Learning and Programming Languages*. 10–19.

[12] Samuel Williams, Andrew Waterman, and David Patterson. 2009. Roofline: An Insightful Visual Performance Model for Multicore Architectures. *Commun. ACM* 52, 4 (2009), 65–76.

[13] Zhiwen Yu et al. 2026. Towards Automated Kernel Generation in the Era of LLMs. *arXiv preprint arXiv:2601.15727* (2026). Section 7: Human–AI Collaboration for Kernel Generation.

[14] Lianmin Zheng, Chengfan Jia, Minmin Sun, Zhao Zhao, et al. 2020. Ansor: Generating High-Performance Tensor Programs for Deep Learning. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*. 863–879.

[15] Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, et al. 2019. Fine-Tuning Language Models from Human Preferences. In *arXiv preprint arXiv:1909.08593*.