2026-Nvidia-iQuHack-Challenge
Product Requirements Document

AutoQurelation

Kevin Joven[1], Abhishek Karna[2], Lang Ji[2], and Matthew Chen[2]

[1]North Carolina State University, Department of Electrical and Computer Engineering, Raleigh, NC, USA
[2]Duke University, Department of Physics, Durham, NC, USA

February 1, 2026

1 Team Roles and Responsibilities

Role	Name	GitHub Handle	Discord Handle
Project Lead	Kevin Joven	`KevinJoven11`	`kevinjoven`
GPU Acceleration PIC	Lang Ji	`lang-ji`	`_langji`
Quality Assurance PIC	Abhishek Karna	`abhishek-karna-duke`	`kronos_3`
Technical Marketing PIC	Matthew Chen	`mattchen10923`	`_thatasian`

2 The Architecture

2.1 Symmetry Structure

A key structural property of the Low Autocorrelation Binary Sequence (LABS) problem is that many distinct bitstrings have equivalent energy due to inherent symmetries of the objective function. In particular, operations such as **global sign inversion (flipping all bits)**, **sequence reversal**, and **related reflection transformations** preserve the aperiodic autocorrelation magnitudes, and therefore yield the same energy value (see our tutorial solution for further details). As a result, the LABS landscape contains large equivalence classes of solutions that differ in pattern but correspond to identical objective quality.

Recognizing these symmetries is directly useful for our optimization pipeline. By factoring out redundant symmetric configurations, we reduce unnecessary exploration of duplicate states, improve the effective diversity of candidate sequences, and accelerate convergence toward globally optimal low-energy solutions. This insight motivates symmetry-aware local search moves and informs the design of our Memetic Tabu Search (MTS) strategy, which ensures computational resources are focused on genuinely distinct regions of the solution space.

2.2 Classical MTS Baseline

We implement a classical Memetic Tabu Search (MTS) solver for LABS that combines population-level exploration with intensive, randomized tabu-based local improvement. Each generation either recombines two parents by single-point crossover or samples an individual from the population, then applies stochastic bit-flip mutation with probability 1/N. The mutated child is refined by a tabu search that explores the single-bit-flip neighborhood with randomized tenure and an aspiration rule permitting tabu moves that improve the global best. The refined child replaces a random population member and the global incumbent is updated when a lower-energy sequence is found. The algorithm terminates on a fixed generation budget or when a known optimal energy is reached. Validation uses exhaustive one-flip local-minimum checks for small N and benchmarking against the well-established LABS optimal-energy table reported in Ref. [8].

2.3 Quantum Enhanced MTS Optimizer

2.3.1 Quantum Algorithm for the "Tutorial" Phase

In the tutorial phase, the quantum component of the workflow is introduced with the goal of establishing conceptual correctness and implementation fidelity rather than achieving asymptotic performance gains. The quantum algorithm is formulated as an adiabatic-inspired variational procedure implemented in CUDA-Q, designed to approximate low-energy configurations of the LABS Hamiltonian for small problem sizes accessible to near-term quantum hardware and classical simulation.

The tutorial algorithm prepares an initial product state that is the ground state of a simple transverse-field Hamiltonian and evolves it toward the problem Hamiltonian encoding the LABS objective. The LABS Hamiltonian is constructed entirely from Pauli-Z operators, reflecting the binary nature of the sequence variables, while the initial Hamiltonian consists of single-qubit Pauli-X terms with a known and easily prepared ground state. The interpolation between these Hamiltonians is discretized into a finite number of steps and implemented as a parameterized quantum circuit using Trotterization.

To mitigate diabatic transitions arising from rapid evolution, the tutorial introduces a first-order counter-diabatic correction derived from the adiabatic gauge potential. Rather than implementing the full, intractable gauge potential, the tutorial follows the approximation regime described in the reference literature, yielding a tractable variational ansatz composed of structured two-qubit and four-qubit interaction blocks. These blocks are implemented explicitly as CUDA-Q kernels and verified through circuit visualization to ensure consistency with the analytical constructions.

The output quantum state is sampled to produce candidate bitstrings, which are evaluated classically using the LABS energy function. In the tutorial setting, these samples are analyzed directly to assess energy distributions and qualitative improvements over random initialization. This establishes the correctness of the Hamiltonian encoding, circuit construction, and sampling pipeline, and provides the foundation for the BUILD-phase integration where quantum-generated samples are used to seed the classical Memetic Tabu Search. Of course, we compare this with Ref. [8] for validation also.

2.3.2 Quantum Algorithm for the "BUILD" phase

The quantum algorithm we chose is a variational algorithm ansatz. The reason is that, in the original QE-MTS paper, Cadavid *et al.* 2025 explain that the scaling of QAOA with 12 layers is approximately $O(1.46^N)$, which is worse than the original work using DCQO [1]. While it is possible to explore more QAOA layers, this would only provide insight into simulations with a high number of layers, which is computationally expensive. Based on Cadavid *et al.* (2025) demonstrating superior circuit-depth scaling (specifically a 6-fold reduction in circuit depth compared to QAOA) we adopt counterdiabatic-enhanced variational optimization over standard QAOA approaches. This is a well-informed scientific choice grounded in recent literature rather than a heuristic risk.

The so-called "expressibility" of an ansatz plays a crucial role in the performance to obtain the solution. If the target solution lies outside the configuration (Hilbert) space reachable by the ansatz, no further performance improvement is possible. Sim *et al.* 2021 presents a collection of different ansätze, along with metrics for expressibility [10]. We use the nearest-neighbor ansatz studied in the paper that has a balance between expressibility and entanglement capabilities. This ansatz consists of single-qubit rotation layers interleaved with nearest-neighbor two-qubit CNOT entangling gates, parameterized by θ_i for $i = 1, \ldots, p$ where p is the circuit depth. We specify that $p = 3$ layers for the BUILD phase to balance expressibility with manageable classical optimization overhead.

The classical optimizer selected for variational parameter optimization is COBYLA (Constrained Optimization BY Linear Approximation), which is gradient-free and well-suited to noisy quantum circuit evaluations. COBYLA requires only function evaluations and is computationally efficient for the moderate parameter counts ($\sim 12-15$ parameters for $p=4$ layers) expected in our variational ansatz. Although there is no formal proof that entanglement can accelerate the simulation of the MTS algorithm, benchmarking different ansätze can offer valuable insights into its performance.

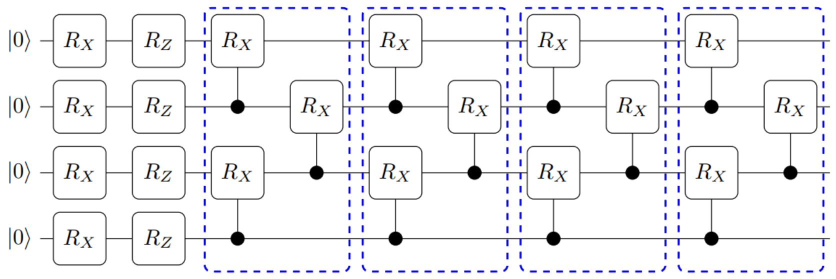

Figure 1: Nearest-neighbor (NN) ansatz. This circuit has a balance between high expressibility and high entanglement. Circuit depth: $p=4$ layers, total parameters: 12-15.

3 The GPU Acceleration Strategy

3.1 Literature Review

Zhang *et al.* 2025 report a massively parallel implementation of memetic tabu search for LABS running on a single **NVIDIA A100 GPU**, explicitly exploiting block- and thread-level parallelism and GPU-friendly binary data structures. They demonstrate up to a $26\times$ speedup versus an analogous 16-core CPU baseline and use the GPU throughput to push to larger problem sizes (up to $N \approx 120$) and obtain improved best-known merit factors for multiple N [11]. One major takeaway to our work is the highest return-on-effort acceleration comes from batch-parallel neighborhood evaluation (scoring many candidate flips in parallel).

Recent tabu search literature in high-performance computing similarly emphasizes restructuring local search to expose massive parallelism—particularly by accelerating the exploration of large neighborhoods on GPUs. Nogueira *et al.* 2025 propose a GPU-based tabu search that harnesses **GPU parallel capabilities** to accelerate neighborhood exploration at scale [3]. Earlier foundational work shows that combining local search (including tabu search) with memetic frameworks is effective for constructing low-autocorrelation sequences and provides baseline algorithmic structures and evaluation protocols [2]. This motivates our use of MTS as a strong classical refinement engine, with GPU acceleration focused on its compute-dominant inner loops.

LABS has also been used to assess quantum optimization methods. Shaydulin *et al.* 2024 study scaling of QAOA time-to-solution (TTS) on LABS and provide evidence that QAOA-based approaches can have favorable empirical scaling compared to strong classical baselines on this benchmark [9]. It inspires our approach to use quantum routines to generate **high-quality seeds** (or guide search), then apply MTS as a refinement stage. On the systems side, NVIDIA's CUDA-Q documentation describes GPU-accelerated circuit simulation and provides explicit guidance for scaling simulations using multi-GPU workflows [7].

3.2 GPUs

We propose a tiered GPU workflow aligned with development velocity, cost control, and performance benchmarking:

1. **L4 for code migration and entry testing.**
   We will use NVIDIA L4 instances for environment bring-up, correctness checks, and rapid iteration on GPU kernels. L4 provides a modern, cost-efficient GPU with 24 GB memory and 300 GB/s memory bandwidth, making it suitable for functional validation and moderate-scale profiling [6]. Development on L4 is estimated at 5-10 hours.

2. **A100 for major performance testing and method comparison.**
   Our primary performance benchmarking and the quantitative comparison of the quantum-augmented

method versus classical baselines will be conducted on NVIDIA A100. This choice is directly supported by the most relevant LABS GPU-MTS study (implemented and benchmarked on A100 with large speedups) [11], and by A100's substantially higher memory bandwidth and larger memory configurations [4].

3. **H100 as an optional extension when credits allow or A100 saturates.**
   If we observe that (i) the solver is demonstrably GPU-bound on A100 (GPU utilization $>80\%$, memory bandwidth saturation $>75\%$ measured via NVIDIA Nsight or nvprof) and (ii) incremental solution quality improvements require substantially more sampling or neighborhood evaluations than A100 can provide within the remaining time/credits, we will escalate to H100. H100 offers significantly higher memory bandwidth and throughput (product configurations report multi-terabyte/s bandwidth) and is best leveraged once kernels are mature and able to keep the device saturated [5].

3.3 Implementation

All numerical experiments in this project were developed and executed within a cloud-based Jupyter notebook workflow, enabling rapid iteration between algorithm design, validation, and benchmarking. Our primary development environment was the *qBraid* platform, which provides an integrated interface for quantum software stacks (including CUDA-Q) alongside classical optimization routines and play as a CPU backend. For GPU backend, we will migrate the implementation from qBraid's default execution environment to the *Brev* GPU platform, which offers accelerated simulation resources suitable for large gate-based quantum circuits and high-throughput sampling.

qBraid itself also provides GPU-enabled execution options, allowing seamless portability between development and accelerated simulation. By leveraging both qBraid and Brev environments, we can achieve an efficient hybrid workflow: qBraid served as a unified research and prototyping interface, while Brev enabled computationally intensive runs on dedicated GPU hardware. This infrastructure ensured that all results reported in Phase 2 are reproducible and scalable, and that the quantum-enhanced routines remain compatible with near-term heterogeneous quantum–classical computing architectures.

3.4 Quantum Acceleration

We will use the GPU primarily to accelerate the quantum seeding stage by running CUDA-Q circuit simulation with a hardware-optimized backend. In particular, the counterdiabatic Trotterized circuits can be executed with high shot throughput on a single GPU, allowing us to sample many candidate bitstrings efficiently. Rather than aiming for asymptotic quantum advantage, we target practical acceleration through high-quality seed generation by increasing the number of measurements and exploring slightly larger N than is feasible on CPU. For larger instances, we can further target CUDA-Q's multi-GPU backend to extend memory limits and improve wall-clock performance.

During the variational optimization phase, we will implement batch parameter evaluation: evaluating at least 10 distinct parameter sets in parallel on the GPU per iteration. Each parameter set is assigned 1000 shots (tunable based on convergence behavior), yielding approximately 10,000-20,000 total quantum samples per optimization step. This is, of course, a rather liberal expectations on our end. But, we remain very optimistic to be able to do this.

3.5 Classical Acceleration (MTS)

The main opportunity for GPU speedup in MTS comes from accelerating repeated energy evaluations during local search. Instead of evaluating neighbor flips sequentially on the CPU, we will implement a batched energy computation using CuPy. The implementation uses binary bit-packed representations for sequences (following Zhang *et al.* 2025), with the following pseudo-structure:

Algorithm 1 GPU Batch Energy Evaluation for MTS

```
Input: Current sequence $s$ (bit-packed on GPU), neighbor flip indices $F = \{f_1, \ldots, f_k\}$ (GPU array),
       autocorrelation precomputed $C_0$ (GPU array)
Output: Energy differences $\Delta E$ for each flip (GPU array)
$S_{gpu} \leftarrow$ copy $s$ to GPU memory
$C_{gpu} \leftarrow$ copy $C_0$ to GPU memory (autocorrelation)
for each flip index $f_i \in F$ (parallelized across GPU threads) do
    $\Delta C_f \leftarrow$ compute affected autocorrelation terms (O(N))
    $\Delta E_i \leftarrow \sum_k (C_k + \Delta C_{f,k})^2 - \sum_k C_k^2$
end
return $\Delta E$ (GPU array)
```

For a neighborhood size of N flips (typical for LABS), this batched evaluation requires memory proportional to $O(N^2)$ for autocorrelation storage and compute time of $O(N^2)$ per batch amortized across N parallel threads. Memory footprint on L4 (24 GB) accommodates $N \lesssim 60$ with room for population; on A100 (80 GB), we support $N \lesssim 120$. Batch size is set to match the full neighborhood (batch size = N), evaluated in a single GPU kernel.

This reduces the dominant inner-loop cost of tabu search and enables more iterations per second, which becomes increasingly important as N grows. Even modest acceleration at this stage allows deeper classical refinement of quantum-generated seeds without changing the overall algorithm structure.

3.6 Hardware Targets

We will develop and validate the full QE-MTS notebook on qBraid using CPU execution to ensure correctness, benchmarking agreement, and reproducibility. For acceleration experiments, we will deploy the same notebook workflow on Brev GPU instances (starting with a single L4, and scaling to A100-class GPUs if needed). This provides a practical near-term path to test larger problem sizes, increase quantum sampling throughput, and reduce the runtime of classical tabu refinement while keeping the implementation simple and portable across platforms.

For GPU correctness validation, we will compare GPU and CPU energy evaluations on small instances ($N \leq 20$). GPU and CPU results must match exactly on CPU; on GPU, we permit relative error tolerance of $\leq 0.1\%$ due to floating-point accumulation order variance in parallel reduction. For larger N, we validate by comparing convergence behavior (best energy found per iteration) to CPU baseline, with acceptance criterion: GPU solution quality within ± 1 energy unit of CPU result at fixed iteration count.

4 Verification Strategies

4.1 Tutorial Phase Tests

In Phase 1 (Tutorial), we established the correctness of the classical Memetic Tabu Search (MTS) baseline before introducing any quantum or GPU enhancements. We first validated the LABS objective implementation on small instances by matching code outputs to hand calculations of

$$E(s) = \sum_{k=1}^{N-1} C_k^2, \qquad C_k = \sum_{i=1}^{N-k} s_i s_{i+k},$$

then benchmarked the solver against published optimal energies for standard test sizes [8]. Representative results are shown below, where we achieved full accuracy and trust in our code.

N	Known Optimum [8]	Our Classical MTS Result
5	2	2.0
7	3	3.0
10	13	13.0
12	10	10.0
15	15	15.0
20	26	26.0

We additionally verified symmetry invariance under global inversion and sequence reversal, and certified that returned solutions are strict local minima by exhaustively evaluating the full single-bit-flip neighborhood. Finally, we tracked basic runtime and evaluation counts to confirm the baseline is fast and reproducible on small-to-moderate sizes ($N \leq 18$). Together, these checks form the regression suite that protects subsequent BUILD-phase development from correctness regressions. We add more unit tests later.

4.2 BUILD Phase Tests

In the BUILD phase, we plan to treat the Phase 1 (Tutorial) checks as the baseline definition of correctness and expand them into an automated regression suite. The core idea is simple: every change must preserve (i) correct LABS energy evaluation, (ii) the expected symmetry invariances, and (iii) the guarantee that the local-search routine returns solutions that are locally optimal under all single-bit flips, while maintaining agreement with known optimal energies on benchmark instances.

To make this repeatable, we convert the previously notebook-based validations into deterministic unit tests with explicit assertions, replacing print-based reporting with pass–fail outcomes that immediately surface regressions. We will implement the following tests in `test.py`:

- **LABS Energy Correctness**: Validate energy computation against hand-calculated results and published optima for $N \in \{5, 7, 10, 12, 15, 20\}$.
- **Symmetry Invariance**: Confirm that global bit inversion and sequence reversal produce identical energy values.
- **Local Optimality**: Verify that MTS solutions are strict local minima under all single-bit flips (exhaustive neighborhood check for $N \leq 20$).
- **GPU vs. CPU Agreement**: Compare GPU batch energy evaluations to CPU baseline; exact match on CPU, relative error $\leq 0.1\%$ on GPU.
- **Convergence Tracking**: Monitor best energy found per iteration; GPU solution quality within ± 1 energy unit of CPU at fixed iteration count.

We complement these unit tests with seeded end-to-end runs: 10 independent executions with different random seeds to validate statistical stability of convergence behavior and cost as we introduce new classical heuristics, GPU kernels, and quantum-circuit components. Automated regression tests are run before and after each major code change to immediately surface regressions.

5 Execution Strategy and Success Metrics

5.1 Workflow

The project workflow is structured to combine targeted use of large language models with domain expertise and manual oversight. Literature review and contextual grounding are performed using *Perplexity Pro Deep Research*, which is used exclusively as a search and summarization tool to identify relevant prior work, extract key results, and cross-check claims against the primary literature. All technical conclusions are validated by direct inspection of the cited sources.

Code development is carried out using a combination of ChatGPT and Perplexity as interactive coding assistants. These tools are used to accelerate drafting, refactoring, and documentation of classical and quantum algorithms, but not as autonomous agents. No AI agents are deployed, and no external API calls are made as part of the workflow. All generated code is reviewed, edited, and validated manually prior to integration.

Algorithmic design decisions, parameter choices, and methodological trade-offs are guided by the team's prior experience with variational and adiabatic quantum algorithms, classical metaheuristics, and hybrid quantum-classical workflows. Where ambiguity arises, preference is given to physically motivated constructions and empirically verifiable behavior over purely heuristic or black-box approaches.

Verification and iteration follow a tight loop: literature-informed design, assisted code generation, manual review and correction, automated testing, and empirical evaluation. This workflow ensures that AI tools function as accelerators of productivity rather than substitutes for scientific judgment, and that all results remain interpretable, reproducible, and technically grounded. **We are very serious about reproducibility and avoiding blackboxes!**

5.2 Success Metrics

Phase 2 of the challenge focuses on scaling and accelerating the workflow through GPU deployment while preserving solution quality relative to established benchmarks. The other major goal is also to access longer sequences N, beyond what we could implement in CPUs locally, in qBraid, and in Google Colab, i.e., $N = 16 - 18$.

We evaluate success using three metrics:

1. *Approximation*: achieve merit factor at least 90% of the best-known solution for sequence lengths up to $N = 30$ (or equal to published optimum if $N \leq 20$), measured against known optimal energies from Ref. [8]. Merit factor is defined as $MF = 2N^2 / \sum_{k=1}^{N-1} C_k^2$, and we report the ratio of our MF to the best-known MF.
2. *Speedup*: demonstrate a minimum $5\times$ wall-clock speedup relative to the CPU-only tutorial baseline for small and medium N ($N \in \{20, 30\}$). We would want to implement it with higher speeds, of course.
3. *Scale*: successfully execute the full classical and quantum-enhanced workflow for sequence lengths up to $N = 40$ or more in a stable and reproducible manner.

To meet these targets, Phase 2 will pursue two concrete actions. First, the codebase will be migrated to the NVIDIA Brev portal to access GPU resources and enable iterative, interactive development on accelerated hardware. Second, GPU acceleration will be implemented for both the classical Memetic Tabu Search and the variational quantum ansatz optimizer. For the classical solver, this will include batched population-level energy evaluation, GPU-parallel neighborhood exploration, and optimized autocorrelation computation to reduce per-iteration cost. For the quantum component, GPU-accelerated simulation and batched parameter evaluation will be used to reduce optimization overhead and improve throughput for both small and moderate N.

Verification of the GPU implementation will be performed by reproducing established results reported in Ref. [8]. For small N, GPU and CPU results will be required to match numerically within tolerance (exact on CPU; relative error $\leq 0.1\%$ on GPU). For larger N, solution quality, convergence behavior, and optimality gaps will be compared statistically to the published benchmarks. Speedup claims will be validated using measured wall-clock runtimes under identical experimental conditions.

5.3 Visualization Plan

The following visualizations will be used to evaluate performance, scalability, and the impact of quantum-enhanced seeding.

1. *Energy Distribution vs. Number of Sequences*: energy (y-axis) as a function of the number of sampled sequences (x-axis), illustrating the distribution of solution quality obtained from the population and from quantum-generated samples.
2. *Median Time-to-Solution vs. Problem Size*: median wall-clock time to reach a target energy (y-axis) plotted against sequence length N (x-axis), comparing classical MTS and quantum-enhanced MTS (QE-MTS).

3. *GPU Convergence Rate*: best energy found (y-axis) versus iteration count (x-axis) for GPU-accelerated runs only, comparing convergence behavior for classical MTS and QE-MTS.

These visualizations will be included in the final BUILD-phase deliverable alongside convergence plots showing energy versus iteration for both classical and quantum-enhanced methods.

6 Resource Management Plan

GPU Type	VRAM	Hourly Rate	Estimated Hours	Primary Role
NVIDIA L4	24 GiB	\$0.85/hr	5-10	Debugging and tuning
NVIDIA A100	80 GiB	\$1.44/hr	10-15	Main benchmark runs
NVIDIA H100	80 GiB	\$2.28/hr	0-5 (conditional)	Optional high-end testing

Given the limited cloud allocation, our GPU usage strategy is designed to maximize benchmarking throughput while remaining within strict cost constraints. We have \$20 of compute credit available on the Brev platform and an additional \$30 credit on qBraid, which will serve as a parallel backup environment. On Brev, we will begin with NVIDIA L4 instances (\$0.85/hr, 24 GiB VRAM) for troubleshooting, circuit debugging, and early-stage performance tuning (estimated 5-10 hours). The primary production benchmark runs will be performed on the NVIDIA A100 (80 GiB VRAM) at \$1.44/hr, as this GPU provides the memory capacity needed for larger-N simulations and high-shot sampling in CUDA-Q (estimated 10-15 hours). If additional performance exploration is required and GPU saturation metrics indicate benefit, we will impose a hard cutoff of \$5 on Brev H100 usage. Overall, this tiered plan ensures that development, validation, and scalable experiments can be executed efficiently while maintaining reproducibility across both Brev and qBraid.

7 Conclusion

This Product Requirements Document outlines a technically grounded, GPU-accelerated approach to solving the Low Autocorrelation Binary Sequence problem via quantum-enhanced memetic tabu search. By integrating recent advances from [11] in variational quantum optimization through expressibility with proven GPU acceleration strategies from [1], we target at least a 5× speedup, merit-factor approximation ratio of 90%, and scaling to $N \geq 40$. The hybrid qBraid-Brev workflow, disciplined resource management, and comprehensive verification strategy position the team to execute reliably within the challenge time and budget constraints.

—

Document draft 7:50PM, January 31, 2026

References

[1] Alejandro Gomez Cadavid, Pranav Chandarana, Sebastián V Romero, Jan Trautmann, Enrique Solano, Taylor Lee Patti, and Narendra N Hegade. Scaling advantage with quantum-enhanced memetic tabu search for labs. *arXiv preprint arXiv:2511.04553*, 2025.

[2] J. E. Gallardo, C. Cotta, and A. J. Fernández. Finding low autocorrelation binary sequences with memetic algorithms. *Applied Soft Computing*, 9(4):1252–1262, 2009.

[3] Bruno Nogueira et al. Gpu tabu search: A study on using gpu to solve massive instances of the maximum diversity problem. *Journal of Parallel and Distributed Computing*, 2025. In press.

[4] NVIDIA. Nvidia a100 tensor core gpu, 2020. Product specifications page. Accessed: 2026-01-31.

[5] NVIDIA. Nvidia h100 tensor core gpu, 2022. Product specifications page. Accessed: 2026-01-31.

[6] NVIDIA. Nvidia l4 tensor core gpu, 2023. Product specifications page. Accessed: 2026-01-31.

[7] NVIDIA. Cuda-q documentation: Multi-gpu workflows, 2024. Accessed: 2026-01-31.

[8] Tom Packebusch and Stephan Mertens. Low autocorrelation binary sequences. *Journal of Physics A: Mathematical and Theoretical*, 49(16):165001, 2016.

[9] Ruslan Shaydulin et al. Evidence of scaling advantage for the quantum approximate optimization algorithm on a classically intractable problem. *Science Advances*, 10(21):eadm6761, 2024.

[10] Sukin Sim, Peter D Johnson, and Alán Aspuru-Guzik. Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms. *Advanced Quantum Technologies*, 2(12):1900070, 2019.

[11] Z. Zhang, J. Shen, N. Kumar, and M. Pistoia. New improvements in solving large labs instances using massively parallelizable memetic tabu search. *arXiv preprint arXiv:2504.00987*, 2025.