PRD: A custom and GPU accelerated hybrid quantum-classical workflow for the LABS problem

Team: LABSrats

February 1, 2026

Abstract

The goal is to identify and implement a different quantum algorithm (e.g., QAOA, VQE, a custom Ansatz, etc.) or a variation of the Counteradiabatic approach to replace the Counterdiabatic approach from Milestone 1 using CUDA-Q. Additionally, we will design a gpu based workflow that accelerates both the Quantum Algorithm (using CUDA-Q) and the Classical MTS on NVIDIA GPUs.

GitHub Forked Repository: https://github.com/RicardoGaGu/2026-NVIDIA

Contents

1 Technical Roles

Role	Name	GitHub Handle	Discord Handle
Project Lead (Architect)	Ricardo García	@RicardoGaGu	@ricardogarciagutierrez
GPU Acceleration PIC (Builder)	Raul Martinez Pavon	@QuantumRaul	@linoobx
Quality Assurance PIC (Verifier)	Stelian Adrian Stanci Sipos	@UO277653	@adrianstanci
Technical Marketing PIC (Storyteller)	Justins Pulpakunnel Varkey	@justinspv24	@justinspv24

2 Research and plan

2.1 Overview of the custom quantum enhanced MTS solver

We describe a couple quantum algorithmic approaches that is distinct from the tutorial's digitized counterdiabatic approach, as well as planning GPU acceleration for the quantum components.

2.1.1 A sample based CVaR VQA for MTS

We propose a variational quantum seed generator inspired by a sampling-based CVaR-VQA workflow [1]. The method can be decomposed into:

1. Repeatedly samples candidate LABS bitstrings from a parametrized quantum circuit
2. Evaluates the LABS energy classically
3. Aggregates low-energy samples via a CVaR objective
4. Updates parameters of a classical cost function via classical optimizer (e.g. COBYLA, SPSA) to bias the distribution toward better quantum based seeds.

The CVaR-VQA scheme is not restricted to QUBO; it samples bitstrings and evaluates the cost classically. This is critical: for LABS, we can compute the LABS energy (sum of squared off-peak autocorrelations) classically on the sampled bitstrings without embedding the full 4-local Hamiltonian into the circuit. The LABS objective can be expressed as a classical cost function over bitstrings, so its corresponding Hamiltonian is diagonal in the computational basis, typically expressed as a sum of multi-Z pauli products (including 4-local terms due to squared autocorrelations).

The quantum stage thus serves as a seed-quality amplifier, and its output is passed to MTS, mirroring the paper's quantum–classical memetic MTS pipeline. Practically, we aim to bias the sampling distribution toward low-energy sequences so that MTS starts from a better pool, rather than claiming standalone quantum optimality, like the original approach.

We can consider 3 main type of benefits:

- **Classical Cost function evaluation**: LABS cost is efficient to compute classically for each sampled bitstring; this fits the sampling-based VQA model. Compared to other quantum optimization approaches, the sampling-based VQA offers some important benefits, since the calculation of the cost is performed classically and its applicability is not restricted to QUBOs, but rather allows for arbitrarily complex objective functions such as higher-order and non-polynomial objectives.
- **Circuit depth vs. seed quality**: As in the paper, harder-to-simulate ansätze can yield better convergence than easy-to-simulate circuits. However, with a shallow LABS-inspired ansatz. Symmetries could help.
- **Noise robustness**: CVaR aggregation is known to be more robust to noise because it focuses on the low-energy tail of samples, which aligns with the seed-generation goal.

This heuristic quantum VQA might show competence against other quantum seeded MTS approaches, and it delegates the evaluation cost to the classical computer and variationally trains a parametrized quantum circuit to obtain good seeds for MTS. If trainability issues can be overcome, robustness to noise is strong and measurements/shots budgets can be kept low, the approach can be promising in a quantum-classical workflow, as demonstrated for portfolio optimization.

2.1.2 Pauli Correlation Encoding (PCE) based VQA for MTS

Pauli Correlation Encoding (PCE) is a qubit-efficient variational framework that has already been applied to LABS [2]. It encodes many binary variables into expectation values of Pauli correlators, allowing optimization with fewer qubits while retaining a variational, sampling-based workflow. This also makes PCE well-suited for quantum seed generation.

PCE might emerge as a practical alternative to counterdiabatic or adiabatic style approaches when qubit count is the dominant constraint. In [2], they used numerical simulations to show that the PCE-based LABS solver has the runtime scaling of 1.330^N for even and 1.326^N for odd instances, which improves over the pure QAOA scaling and is marginally better even than the state-of-the-art classical heuristics such as Tabu Search, but has not been compared to MTS directly.

However, some engineering effort to finetune the PCE around variational asantz approach and the possibility of blending PCE derived samples seeds into MTS runs might render the approach competent with the best classical solver fro LABS and the quantum counter adiabatic based MTS solver. Let's analyze some of the benefits of the approach:

- **Directly feasible:** This method is explicitly applied to LABS in the paper, including the long-range 4-local structure (Bernasconi model).
- **Qubit-efficient:** PCE encodes N binary variables in poly$(\log N)$ qubits (polynomial reduction), enabling larger N on near-term devices or simulators.
- **Seed generation fit:** The output is a variationally optimized distribution over bitstrings, which can be used as quantum seeds for MTS, similar to your pipeline.
- **Sampling-based training:** The approach relies on estimating correlators via shots; it is compatible with noisy devices and with GPU-accelerated simulators. Instead of a trotterized evoloution, we still need to train the VQA!

We can compare this approach against other typical quantum optimization pipelines:

Relative to counterdiabatic or adiabatic HUBO/QUBOs (e.g., DCQO / AQC): PCE reduces qubit count dramatically, whereas adiabatic/QUBO mappings typically require one qubit per variable (or more after reductions), and reported time-to-solution scaling is competitive with classical Tabu search, suggesting strong seed generation without deep adiabatic schedules.

Relative to standard gate-based seed generation (e.g., QAOA or generic ansätze): PCE targets fewer qubits for the same N, which is often the binding constraint, and its correlator encoding is designed to mitigate barren-plateau trainability issues.

2.2 Milestones and Experimentation Plan

2.2.1 Experiments Strategy

We will validate whether any of the proposed quantum-seeded MTS improves classical performance in practice, without claiming quantum optimality. We compare two quantum-seed generators—PCE and CVaR-VQA and measure downstream impact on MTS across a possible range of LABS sequence lengths N = 25 to 100. Ideally, for each N and method, we would run R = 30–50 independent trials with different random seeds. Lower budgeting might apply given priorities.

Methods to be compared:

- **Baseline:** Classical MTS with random initialization.
- **PCE-Seeded MTS:** Use PCE variational training to generate a seed pool, then run MTS. PCE is already applied to LABS and shows competitive scaling and qubit efficiency.
- **CVaR-VQA-Seeded MTS:** Sampling-based VQA optimized via CVaR; generate seeds, then MTS. This mirrors a quantum-classical pipeline where the quantum stage improves the seed distribution.
- **Optional Control:** DCQO-seeded MTS (from the QE-MTS framework) for comparison to prior quantum seeding.

Primary metrics for success We will report the median time-to-solution (TTS) required to reach a target energy at each N. The target is the best-known energy for small N, or within $X\%$ of the best-found across methods for larger N. We also report the median best energy within a fixed wall-clock budget and the success probability, defined as the fraction of trials that reach the target within budget. These metrics follow the scaling and TTS evaluation style used in QE-MTS. [**?**]

Secondary metrics We can compare the energy distribution of quantum seeds against random seeds, the MTS convergence speed (iterations to target), and the compute cost per seed (shots or GPU time) to quantify the seed-quality tradeoff.

2.2.2 Milestones (both CPU/GPU)

1. **CPU baseline:** Run classical MTS with random initialization for $N = 25$–100; log median TTS, median best energy, and success probability.
2. **CPU validation:** Verify energy computation and symmetry tests on small N; lock baseline parameters and budgets.
3. **GPU seed generator (PCE):** Train PCE on GPU simulator; generate fixed-size seed pools; record seed energy distributions and compute cost.
4. **GPU seed generator (CVaR-VQA):** Train CVaR-VQA on GPU simulator; generate seed pools under matched budgets.
5. **Hybrid runs:** Feed PCE and CVaR-VQA seeds into MTS; measure the same metrics as baseline.
6. **Comparison report:** Summarize CPU vs GPU results and seed quality impacts using median TTS and success probability plots.

Note: Not all custom approaches might be tested or implemented.

We do not claim quantum optimality. We validate whether quantum-seeded MTS yields (i) better median energy, (ii) faster median TTS, and (iii) higher success probability than classical MTS alone, in a scaling sense. It will be interesting to compare to the counteradiabatic approach in terms of resources, quality of solution and time to solution.

2.2.3 Basic unit tests

The following checks will be implemented as automated assertions in `tests.py`:

1. **Energy function correctness (hand-computed + brute-force):**
   - `assert labs_energy([1, -1, 1]) == 5`
   - `assert labs_energy([1, 1, 1, 1]) == 14`
   - `assert labs_energy([1, -1]) == 1`
   - `assert labs_energy([1]) == 0`

   Additionally, brute-force exhaustive search for $N = 3$–7 asserts that the minimum energy matches known optima (E^*: 1, 2, 2, 7, 3). This validates both the energy function and any GPU-accelerated variant (e.g. CuPy-based batch energy).

2. **Symmetry invariance (all four LABS symmetries):** For random sequences of length $N \in [5, 20]$, assert:
   - `assert energy(s) == energy(-s)` (negation)
   - `assert energy(s) == energy(s[::-1])` (reversal)
   - `assert energy(s) == energy(-s[::-1])` (negation + reversal)
   - `assert energy(s) == energy(alternating_modulate(s))` ($s_i \to (-1)^i s_i$)

Tested over 10+ random sequences. Applied to both the NumPy and CuPy implementations to ensure the GPU port preserves physics.

3. **MTS end-to-end convergence:** Run the custom MTS (including any GPU-accelerated components) with multiple seeds for $N = 5, 6, 7$. Assert that the best energy equals the brute-force optimum.

4. **Custom quantum ansatz output validation:**

   - For CVaR-VQA: Assert the CVaR loss decreases over optimizer iterations (convergence sanity check) and that trainability issues are not dominant.

5. **Custom Quantum-seeded MTS vs CD Quantum-seeded MTS vs. random-seeded MTS:** For $100 \geq N \geq 25$, assert that the quantum-seeded population (from CVaR-VQA or PCE) can show competitive scaling in with classical MTS and CD quantum seeded MTS. We might not sweep all N, but select the most promising approach to compute its scaling across many runs.

2.3 GPU Planning for numerical simulations

2.3.1 Quantum Acceleration (CUDA-Q)

- **Strategy:** We will maximize simulation thoughput by migrating the main CPU-based kernet to GPU acceleration, leveraging CUDA-Q's nvidia target.

- **Algorithm & Qubit Mapping:** We will utilize a Direct Ising Model mapping where the qubit count scales linearly with the sequence length.

  - **Implication:** Unlike compact encodings, this approach places a strict memory requirement on the simulation. We will focus on optimizing the CVaR-VQA ansatz efficiency toi generate high-quality seeds within the memory limits of the GPU state vector simulation.

- **Backend Implementation:** We will explicitly set `cudaq.set_target("nvidia")` to utilize StateVector simulation backend over the GPU instead of the default CPU aproach. This enables the use of cuQuantum libraries to handle the exponential complexity of the chosen mapping.

2.3.2 Classical Acceleration (MTS)

- **Strategy:** By rewriting the energy function using the package `CuPy` instead of `NumPy`, we will eliminate the primary bottleneck of the classical MTS, the iterative evaluation of neighbour energies

  1. **Vectorization:** Instead of evaluating one by one all the N neighbours of a single candidate solution we will stack the entire neighborhood into a single GPU matrix.
  2. **Parallel Computing:** We will use `CuPy`'s capablilities to calculate the autocorrelation energy of multiple bitstrings simultaneously in a single kernel launch, transforming an $\mathcal{O}$(N) sequential process into a massively parallel operation.

2.3.3 Hardware Requirements

We will be using two hardware instances to validate and test the algorithm for larger bitstring sizes.

- **Brev L4:** We will be using this cost-effective instance for development, refining and initial validation of the VQA/MTS pipeline for small instances ($N \leq 30$)

- **Brev A100:** It will be reserved for the algorithm benchmarking runs.

  1. **Direct Mapping:** Memory usage will become the bottleneck saturating the GPU with the state vector with $N \approx 38$. With this approach the 880GB VRAM is critical to push the simulation.

2.3.4 Budget Estimation

Hardware selection follows a strict cost-efficiency rule to fit in the budget within the timeline.

- **Brev L4:** Used for logic porting and verification for small N.
  - Estimated cost/time ratio $\approx \$0.85/hr$.
  - Estimated time usage $\approx 3hours$
  - Estimated cost of the instance $\$0.85/hr \cdot 4 \approx \$2.55$
- **Brev A100:** Reserved for the final benchmark runs.
  - Estimated cost/time ratio $\approx \$3.90/hr$.
  - Estimated time usage $\approx 3.5hours$
  - Estimated cost of the instance $\$3.905/hr \cdot 4 \approx \$13.65$
- **Total budget Spent:** $\approx \$16.20$ which leaves us with a buffer of $\approx \$3.80$ for idle usage and some retries.

References

[1] Gabriele Agliardi, Dimitris Alevras, Vaibhaw Kumar, Roberto Lo Nardo, Gabriele Compostella, Sumit Kumar, Manuel Proissl, and Bimal Mehta. Portfolio construction using a sampling-based variational quantum scheme. *arXiv preprint arXiv:2508.13557*, 2025.

[2] Marco Sciorilli, Giancarlo Camilo, Thiago O. Maciel, Askery Canabarro, Lucas Borges, and Leandro Aolita. A competitive nisq and qubit-efficient solver for the labs problem. *arXiv preprint arXiv:2506.17391*, 2025.