Product Requirements Document: Project GQE-MTS

TEAM QAT

February 1, 2026

Hybrid Generative Quantum-Enhanced Memetic Tabu Search for LABS

Team Name: QAT
GitHub Repository: `https://github.com/rnsln/iquhack2026-nvidia`
Github Profiles:
Hatice Boyar: https://github.com/dhaticeboyar
Eren Aslan: https://github.com/rnsln
Hüseyin Umut Işık: https://github.com/HUmutI
Web Page: COMING SOON...

1 Team Roles & Responsibilities (PICs)

- **Project Lead (Architect): HATICE BOYAR and CHANG JEN YU**
Responsible for the algorithmic "North Star," high-level system integration, and defining the Hamiltonian. Ensures the theoretical validity of the ansatz.

- **GPU Acceleration PIC (Builder): HUSEYIN UMUT ISIK**
Lead for the CUDA-Q 0.13.0 implementation and the CuPy-based parallel evaluation engine. Responsible for optimizing memory coalescing in CUDA kernels and "Zombie Instance" prevention on Brev.dev.

- **Quality Assurance PIC (Verifier): ILAYDA DILEK**
Owner of the Verification Strategy. Responsible for the automated unit test suite and Python library `Hypothesis` to guard against AI hallucinations and ensure physical consistency.

- **Technical Marketing PIC (Storyteller): EREN ASLAN**
The data analyst. Responsible for converting raw logs into performance visualizations (Speedup/Convergence plots). **Key Initiative:** Developing a **Live Interactive Web Dashboard** (Node.js + Express.js + HTML). This platform will feature

dynamic charts showing the real-time descent of the MTS algorithm and 3D visualizations of the Quantum Energy Landscape, providing an immersive storytelling experience for the judges.

2 The Architecture: Hybrid GQE-MTS

Owner: Project Lead - CHANG JEN YU and Hatice Boyar

Methodology: Hybrid GQE-MTS Architecture

Our approach synergizes the **global exploration capabilities of Quantum Computing (GQE)** with the **local exploitation capabilities of Classical Heuristic Algorithms (MTS)[2]** to solve the highly complex Low Autocorrelation Binary Sequences (LABS)[3] problem.

Evolution and Rationale

Initially, our research utilized a GPU-accelerated QE-MTS approach. However, we observed that this method occasionally yielded inconsistent results due to stochastic errors. While we considered variational methods such as VQE[4] or QAOA[5] as alternatives, these approaches suffer significantly from the **"Barren Plateau" problem[6]** in high-dimensional optimization landscapes.

Consequently, we pivoted to **Generative Quantum Eigensolvers (GQE)[1]** augmented by **Generative AI** for circuit synthesis. This approach avoids the pitfalls of traditional variational parameter optimization by guiding the circuit generation process.

Key Innovations

We introduce two distinct novelties to adapt GQE for the LABS problem:

1. **Transfer Learning[7] for GQE Acceleration:**
   Standard GQE requires substantial computational resources. We integrated **Transfer Learning** to accelerate prediction. By training on smaller sub-problems and transferring the geometric insights to larger instances, we significantly reduce the training latency.
2. **Non-Chemical Domain Adaptation:**
   Unlike standard GQE rooted in quantum chemistry, we treat LABS as a distribution problem. We construct our sampling space using **2-body and 4-body operators** derived from quantum annealing principles.

The methodology is divided into three distinct phases:

Phase 1: Quantum Generative Optimization (GQE Phase)

The goal of this phase is not to find the perfect solution immediately, but to **locate the "Basin of Attraction"** where the good solutions reside.

1. **Physical Modeling:** We map the LABS autocorrelation energy into a Quantum Hamiltonian (H).

2. **Operator Pool Design:**

   - We utilize problem-specific operators (two-body and four-body terms) derived from the interaction graph.
   - **Hierarchical Parameters:** We employ a decaying parameter strategy ($\pi \to \pi/8$). This allows the GQE to perform large geometric rotations (coarse tuning) first, followed by fine quantum interference adjustments (fine tuning).

3. **Transfer Learning Strategy (Input/Output Adaptation):**
   To solve the target problem size (N_{target}), we employ a specific training protocol on smaller instances (N_{train}):

   - **Zero-Padding Input:** We define the model architecture based on the maximum target size (N_{target}). When training on smaller sub-problems ($N_{train} < N_{target}$), the input vectors are **zero-padded** to match the dimensionality of N_{target}.
   - **Output Truncation:** During the training process, the model generates a circuit for the full dimension. However, for loss calculation, we **truncate or mask the output** (the extra qubits/gates beyond N_{train}) to evaluate the energy only within the valid subspace of the current training instance.
   - This allows the generative model to learn the fundamental geometric features of LABS interactions on small N and transfer these weights to larger N without retraining from scratch.

4. **GQE Engine Execution:**

   - The engine samples operators from the pool using the pre-trained/transferred model.
   - It minimizes the energy expectation value $\langle H \rangle$.
   - **Result:** An optimized **Quantum Ansatz** where the wavefunction is concentrated on low-energy states.

Phase 2: Bridging & Filtering Phase

This phase acts as the interface between the quantum and classical worlds, converting "quantum probabilities" into "high-quality seeds."

1. **Sampling:** We execute 1000 shots on the optimized circuit to obtain binary sequences (Bitstrings).

2. **Energy Validation:** Convert bitstrings back to the spin format and use a classical CPU to calculate the exact LABS energy.

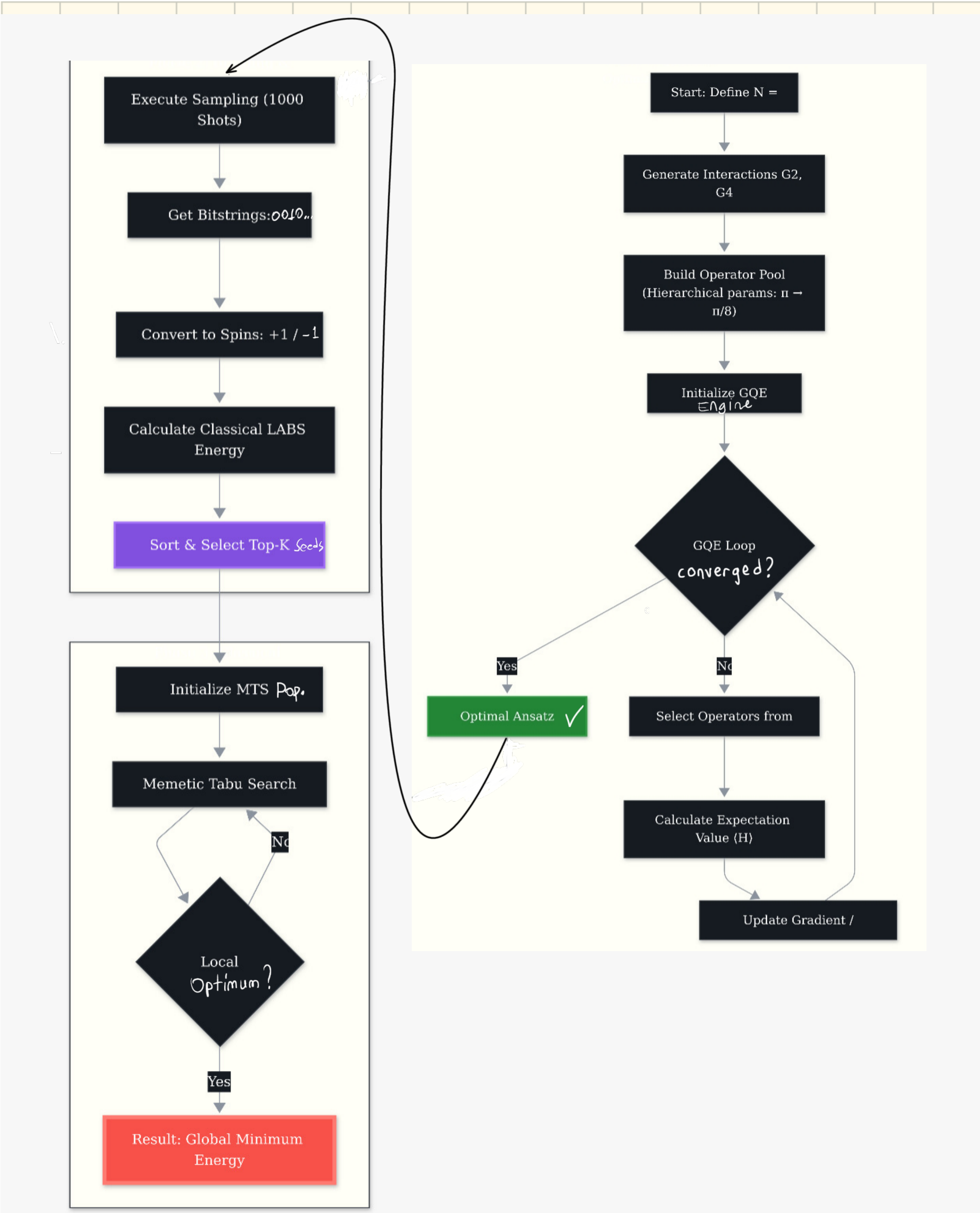
3. **Elitism Filtering:** From the 1000 samples, we select only the top K (e.g., 20) unique sequences with the lowest energy. These become our **"Golden Seeds."**

Phase 3: Classical Memetic Search (MTS Phase)

This phase leverages cost-effective classical computing to perform the final optimization sprint.

1. **Seed Injection:** The "Golden Seeds" are injected as the **Initial Population** for the MTS.
2. **Tabu Search:** The MTS performs bit-flips within the neighborhood of these seeds, utilizing a Tabu list to prevent cycling back to previous states.
3. **Convergence:** Since the starting points are already deep within the energy valleys, the MTS can slide down to the **Global Minimum** extremely fast.

System Architecture Flowchart



Why This Method Works

1. **Winning at the Starting Line:**

   - Pure random search is like dropping onto the Himalayas blindly; it is statistically unlikely to land near the peak of Mt. Everest.
   - Our method uses Quantum Radar (GQE) to scan for "where the mountain is" and drops the climbing team (MTS) halfway up the slope.

2. **Physics-Aware:**

   - The Operator Pool is not random; it is strictly defined by the structure of the LABS problem. This means every step the quantum circuit takes is a valid physical evolution, making it far more efficient than blind parameter optimization.

3. **Fault Tolerance:**

   - Even if the GQE does not find the perfect Ground State, as long as it finds a sufficiently low excited state, the MTS can easily complete the "last mile" of optimization.

3 The Acceleration Strategy

Owner: GPU Acceleration PIC - HUSEYIN UMUT ISIK

Quantum Acceleration (CUDA-Q)

- **Strategy:** We utilize the `nvidia` backend in CUDA-Q 0.13.0 for high-performance state-vector simulation.
  - **Multi-GPU Scaling:** For sequence lengths $N > 30$, we leverage `nvidia-mgpu` on Brev to distribute the large state-vector (2^N amplitudes) across multiple L4/A100 GPUs using MPI decomposition.
  - **Circuit Optimization:** We use `cudaq.optimize` passes to fuse single-qubit gates and reduce circuit depth before execution.

Classical Acceleration (MTS)

- **Strategy:** The bottleneck of MTS is evaluating the N single-flip neighbors.
  - **Delta-Evaluation Optimization:** Instead of recalculating energy from scratch ($O(N^2)$), we compute the change in energy (ΔE) for a bit-flip, which reduces complexity to $O(N)$.

- **CuPy Parallelization:** We implement a custom CUDA kernel via CuPy to evaluate all possible N flips and Tabu criteria in parallel, achieving a massive speedup over serial CPU iterators. We tune GPU performance by adjusting kernel granularity (threads per block and grid dimensions) to maximize occupancy and minimize launch overhead.

- **Hardware Targets:**
  - **Dev:** qBraid (CPU) for logic and unit testing.
  - **Production:** Brev (A100-80GB) for final benchmarks on bigger N values with different hyperparameters like FP32, FP64, MGPU on/off, etc.

4 The Verification Plan

Owner: Quality Assurance PIC - ILAYDA DILEK

Core Correctness Checks

- **Check 1 (Symmetry & Invariants):**
  - **Reversal Symmetry:** $E(S) = E(S_{\text{reversed}})$.
  - **Negation Symmetry:** $E(S) = E(-S)$.
  - **Energy Bounds:** Ensure computed energy does not violate theoretical lower bounds for LABS.
- **Check 2 (Ground Truth Calibration):**
  - **Small N:** For $N = 7$, must return $E = 3$.
  - **Medium N:** For $N = 20$, must target known minimum $E = 16$ (Merit Factor ≈ 6.0).
  - Verify results against known Barker Codes for applicable lengths.
- **Check 3 (Operator Pool & Gradients):**
  - Unit tests to ensure the 4-qubit R_{ZZ} chains in the DCQO circuit correctly preserve parity.
  - Finite-difference checks to verify analytic gradients used in the GQE loop.

AI Hallucination Guardrails

- We use a "Test-Driven Prompting" strategy: AI agents are given the test file before the implementation request. Any code that does not pass the automated `tests.py` suite is immediately rejected for refactoring.

5 Execution Strategy & Success Metrics

Owner: Technical Marketing PIC - EREN ASLAN

Agentic Workflow

- **Orchestration:**
  - **Agent Alpha (Architect):** Decomposes tasks into atomic issues (e.g., "Implement Interaction Graph Generation").
  - **Agent Gamma (QA):** Reviews PRs, analyzes `pytest` logs, and spots "silent failures" where code runs but produces physics-violating results.
- **Vibe Coding Tip:** We will keep a live "Vibe Check" log to record where the AI agents struggled with CUDA-Q syntax vs. classical Python logic.

Success Metrics

- **Target Hit Rate:** Achieve the global minimum for $N = 27$ in under 120 seconds.
- **Speedup:** Target a 20x speedup on the Brev A100 vs. the qBraid CPU baseline.
- **Approximation Ratio:** Maintain an energy ratio > 0.9 relative to best-known values for $N = 40$.

6 Resource Management Plan

Owner: GPU Acceleration PIC

- **Credit Allocation Strategy (Brev.dev):**
  - **Prototyping Phase:** Initial development, debugging, and unit testing are performed on cost-efficient GPU instances using small-to-moderate problem sizes. During this phase, all code changes are validated through the automated `tests.py` suite to prevent unnecessary GPU usage caused by incorrect or unstable implementations.
  - **Benchmarking Phase:** High-performance GPU instances are reserved exclusively for large-scale production runs and benchmarking experiments (e.g., $N \geq 30$), after functional correctness and stability have been verified.
  - **Operational Buffer:** A portion of available credits is intentionally reserved to accommodate re-runs, profiling, debugging, and unexpected operational overhead.

- **Zombie Prevention and Resource Guardrails:**
  - **Automated Monitoring:** A lightweight background script monitors GPU utilization via `nvidia-smi` at regular intervals to detect idle or stalled instances.
  - **Policy Enforcement:** If sustained low utilization is detected over a predefined time window, the instance is automatically terminated to prevent unintended credit consumption.

References

[1] Nakaji, Kouhei, et al. *The generative quantum eigensolver (GQE) and its application for ground state search.* arXiv preprint arXiv:2401.09253 (2024).

[2] Gomez Cadavid, A., Chandarana, P., Romero, S. V., et al. *Scaling advantage with quantum-enhanced memetic tabu search for LABS.* arXiv preprint arXiv:2511.04553 (2025).

[3] Packebusch, T., and Mertens, S. *Low autocorrelation binary sequences.* Journal of Physics A: Mathematical and Theoretical, 49(16), 165001 (2016). DOI: 10.1088/1751-8113/49/16/165001.

[4] Tilly, J., Chen, H., Cao, S., et al. *The Variational Quantum Eigensolver: A review of methods and best practices.* Physics Reports, 986, 1–128 (2022). DOI: 10.1016/j.physrep.2022.08.003.

[5] Farhi, E., Goldstone, J., and Gutmann, S. *A Quantum Approximate Optimization Algorithm.* arXiv preprint arXiv:1411.4028 (2014). URL: https://arxiv.org/abs/1411.4028.

[6] Larocca, M., Thanasilp, S., Wang, S., et al. *Barren plateaus in variational quantum computing.* Nature Reviews Physics, 7(4), 174–189 (2025). DOI: 10.1038/s42254-025-00813-9.

[7] Zhuang, F., Qi, Z., Duan, K., et al. *A Comprehensive Survey on Transfer Learning.* Proceedings of the IEEE, 109(1), 43–76 (2020). arXiv preprint arXiv:1911.02685.