Q-DELIBERATE

A Quantum Deliberation Fabric for Frontier Large Language Models

A falsifiable hybrid quantum-classical architecture for reasoning search, retrieval, planning, routing, scientific synthesis, and agentic computation

Scientific and Engineering Research Proposal

Version 1.0 - September 2026

Status: speculative research hypothesis; not an established quantum advantage

Central proposition: quantize deliberation, not the neural network.

Abstract

Q-Deliberate is a proposed architecture for connecting frontier large language models (LLMs) to quantum processors without attempting to execute the Transformer itself on a quantum computer. The theory begins from a negative result: dense neural operations such as matrix multiplication, attention, embedding transport, and layer-by-layer inference are poorly matched to present quantum hardware because arbitrary classical state preparation, repeated measurement, quantum-control latency, noise, and circuit-depth constraints can erase any notional advantage. The surviving hypothesis is therefore narrower and more falsifiable. A conventional LLM remains on classical AI accelerators; selected high-level decisions are compiled into a compact Quantum Deliberation Intermediate Representation (QDIR), representing constrained discrete search, optimization, or sampling problems. A Hardness and Value Router then chooses among CPU, GPU, simulated-QPU, physical-QPU, and future fault-tolerant-QPU backends. Quantum outputs are never treated as authoritative answers; they are candidate configurations that pass through classical verification before they can affect generation.

Three explicit falsification attempts are built into the theory. The first rejects direct hidden-state amplitude encoding as the core interface because data loading can dominate the computation. The second rejects token-by-token QPU calls because communication, queueing, compilation, shot repetition, and measurement latency can exceed autoregressive inference budgets. The third rejects any assumption of automatic quantum superiority because strong classical solvers may dominate every useful problem family. The architecture is amended after each failure. The final theory claims only that there may exist naturally occurring LLM deliberation subproblems for which quantum processors eventually achieve a better quality-cost-latency-energy Pareto frontier than the strongest classical alternatives, and that better solutions to those subproblems may translate into measurable gains in end-to-end model capability. Both parts must be demonstrated experimentally.

The proposed program can begin immediately using open-weight or API-hosted frontier models, GPU quantum simulation, multiple classical solvers, and cloud-accessible QPUs. A rigorous minimum viable program is estimated at roughly US$2-5 million over 12-18 months; a serious three-year multi-backend program is estimated at roughly US$22-61 million. A larger fault-tolerant transition program could exceed US$50-150 million, excluding fabrication of a quantum computer and excluding training a new frontier foundation model from scratch. These are planning ranges, not vendor quotations.

Executive Findings

Finding	Result
Where quantum computation belongs	Structured search, optimization, sampling, planning, and combinatorial selection generated by the LLM - not dense Transformer layers.
Core interface	QDIR: a compact, auditable representation of binary/integer decision variables, objectives, constraints, metadata, deadlines, and verification requirements.
Reliability architecture	Classical solver and verifier run in parallel or fallback mode; QPU failure cannot block normal service.
Scientific success criterion	A statistically significant end-to-end model improvement or cost/latency/energy improvement under matched budgets, after defeating strong classical baselines.
Most likely early win	Complementary candidate diversity or a narrow hybrid-utility region, not a universal exponential speedup.
Long-term thesis	As AI inference becomes more deliberative and agentic, heterogeneous GPU/CPU/QPU systems may outperform purely classical stacks on selected cognitive search workloads.

Contents and Reading Guide

This paper is organized as an implementation document rather than a purely conceptual essay. Sections 1-5 define the theory and computational model. Sections 6-9 record the falsification process. Sections 10-17 specify algorithms, QDIR, routing, verification, data, and software. Sections 18-23 define experimental design and measurement. Sections 24-31 specify infrastructure, personnel, security, schedule, cost, procurement, and risk. Sections 32-36 describe future fault-tolerant extensions, distillation, and long-range research consequences. Appendices provide example schemas, benchmark templates, acceptance gates, and a reference project specification.

1. Problem Statement and Design Objective

Frontier LLMs are increasingly limited not only by the cost of producing a token but by the amount of useful search they can perform before committing to a token, action, plan, tool invocation, retrieved memory set, or scientific hypothesis. The research objective is to determine whether quantum processors can become a useful accelerator for this deliberative computation while preserving the strengths of classical neural inference.

The project deliberately excludes a much stronger and less defensible claim: it does not claim that a QPU should replace GPUs for matrix multiplication or execute the full neural network. The primary system boundary is therefore between neural inference and structured decision search.

LLM -> QDIR Compiler -> Hardness/Value Router -> {CPU, GPU, QPU} -> Verifier -> LLM

Figure-equivalent architecture expressed as a processing chain.

1.1 Formal research question

Let P_n denote a distribution of structured deliberation problems naturally produced by an LLM at scale n. Let U_b(P) be a utility function for backend b incorporating verified solution quality, wall-clock latency, financial cost, and energy use. The principal research question is whether there exists at least one relevant family for which a quantum backend eventually becomes preferable after all interface overhead is included.

Exists P_n such that U_Q(P_n) > U_C(P_n) + delta

delta is a preregistered margin that prevents noise-level differences from being described as advantage.

A second, independent requirement is that improved optimization quality must causally improve the LLM task. A quantum solver can be computationally impressive yet useless to language-model capability. Accordingly, both backend advantage and end-to-end AI effect are required.

dS/dQ_D > 0

S is end-to-end model score; Q_D is deliberation solution quality.

2. Why a Quantum Transformer Is Not the Starting Point

Modern Transformer inference is dominated by dense tensor operations, memory movement, KV-cache management, collective communication, and highly optimized GPU kernels. Quantum devices offer a different computational model. A mathematical representation that uses few qubits to index a large vector does not imply that an arbitrary classical vector can be prepared for free, nor that all of its components can be read out efficiently. For an LLM, repeatedly encoding hidden states, running circuits, and measuring results at every layer or token introduces an unfavorable interface boundary.

2.1 State preparation bottleneck

$$|h\rangle = \frac{1}{\|h\|}\sum_i h_i |i\rangle$$

Although the index register is logarithmic in vector dimension, arbitrary preparation of |h> from classical memory can require substantial work. If state preparation scales with the amount of classical information being transferred, the apparent advantage of manipulating the state after loading may be overwhelmed by loading itself.

2.2 Measurement bottleneck

A quantum state cannot generally be read out as an arbitrary full classical vector in a single operation. If an LLM needs dense layer activations back after every quantum operation, repeated sampling and tomography-like reconstruction would be unacceptable. Q-Deliberate therefore returns sparse candidate configurations or statistics whose classical description is intentionally small.

2.3 Latency and control

Interactive autoregressive inference requires predictable service-level latency. A remote or cryogenic QPU adds network transmission, circuit compilation, control scheduling, repeated shots, result collection, and queue variability. Q-Deliberate moves quantum work outside the critical per-token path and permits the LLM to proceed with classical alternatives when quantum results miss a deadline.

3. Core Theory: Quantize Deliberation, Not the Network

The central abstraction is a latent structured decision variable z. Given context x, a conventional model computes a representation h = f_theta(x). Rather than sending h directly to a quantum computer, the model or a compiler creates a compact problem P(h) whose variables encode high-level choices. Examples include whether to select a memory chunk, whether a candidate reasoning step belongs in a solution, which expert or tool to use, whether two hypotheses are mutually compatible, or which ordering constraints should hold in a plan.

$$z^* = \arg\min_{z \in \mathcal{Z}} E(z)$$

A quantum backend may optimize or sample low-energy states. The result is decoded into candidate structures and then verified classically. The model output can be understood as marginalizing over deliberation structures rather than altering neural weights.

$$P(y \mid x) = \sum_z P_\theta(y \mid x, z)\, P_{\text{delib}}(z \mid x)$$

Q-Deliberate acts on P_delib, not directly on P_theta. That separation permits the underlying LLM to remain unchanged during early experiments, making causal attribution substantially cleaner.

4. QDIR: Quantum Deliberation Intermediate Representation

QDIR is the proposed contract between an LLM system and heterogeneous optimization backends. It is intentionally independent of any single quantum algorithm. A QDIR instance contains decision variables, objective coefficients, hard and soft constraints, problem-family metadata, decoding rules, deadlines, maximum permissible cost, and the verifier to be used. It can be lowered to a QUBO, Ising Hamiltonian, SAT/MaxSAT instance, integer program, graph problem, or other backend-specific form.

4.1 Canonical binary form

$$E(z) = \sum_i a_i z_i + \sum_{i<j} b_{ij} z_i z_j + \lambda C(z), \quad z_i \in \{0, 1\}$$

The first term captures local desirability, the second captures pairwise compatibility or conflict, and C(z) encodes constraints or higher-order penalties after reduction. When mapped to spin variables s_i = 2z_i - 1, the instance can be represented by an Ising-style cost Hamiltonian.

$$H_C = \sum_i h_i Z_i + \sum_{i<j} J_{ij} Z_i Z_j + H_{\text{constraint}}$$

4.2 QDIR design requirements

- Compact: problem representation should be far smaller than the LLM activation tensors from which it was derived.
- Auditable: every variable and coefficient must map back to an understandable candidate, relation, or constraint.
- Backend neutral: no mandatory dependence on QAOA, annealing, a particular vendor, or a particular qubit technology.
- Verifier aware: the representation must specify how returned states will be checked before influencing generation.
- Deadline aware: the scheduler needs a hard cutoff so quantum work cannot stall interactive service.
- Cost aware: QPU use is permitted only within explicit financial and energy budgets.
- Reproducible: problem instances, compiler version, model version, and solver configuration must be hashable and archived.

5. Target LLM Workloads

5.1 Reasoning graph selection

The LLM generates a set of candidate intermediate propositions, calculations, deductions, or tool calls. Variables indicate whether a candidate belongs to the final reasoning structure. Local coefficients reward estimated usefulness. Pairwise coefficients reward mutually supportive steps and penalize contradictions or redundant branches. Constraints encode ordering, budget, tool availability, or required subgoals.

$$E_R(z) = -\alpha \sum_i u_i z_i - \beta \sum_{i<j} c_{ij} z_i z_j + \gamma P(z)$$

5.2 Long-context evidence selection

Retrieval systems normally score chunks individually. QDIR permits global selection of a set in which relevance, redundancy, token cost, source diversity, temporal consistency, and cross-document synergy are optimized jointly. This can reduce the amount of context presented to the LLM while preserving or increasing evidentiary coverage.

$$E_M(z) = -\sum_i r_i z_i + \lambda_d \sum_{i<j} d_{ij} z_i z_j - \lambda_s \sum_{i<j} s_{ij} z_i z_j + P_{\text{token}}(z)$$

$$\sum_i \ell_i z_i \le B$$

5.3 Mixture-of-experts macro-routing

Rather than making an independent expert choice for every token, a macroblock router can optimize assignments jointly over a group of tokens, requests, or reasoning states. The objective can include predicted expert quality, balancing, communication overhead, capacity, and expert diversity. QPU invocation should be batched; token-level invocation is explicitly excluded.

5.4 Agentic tool planning

Agentic systems often face a constrained plan-selection problem: which tools to call, in what order, with what resource budget, and under which dependencies. A graph or binary formulation can encode expected success, cost, latency, prerequisite relationships, mutually exclusive actions, and risk penalties. The QPU proposes candidate plans; execution remains governed by classical policy and verification.

5.5 Scientific hypothesis synthesis

A scientific LLM may generate dozens or hundreds of candidate mechanisms, assumptions, explanatory variables, or model components. QDIR can represent which hypotheses should be combined, which are contradictory, how much evidence each explains, and how much complexity each adds. The QPU is a combinatorial hypothesis composer, not an oracle of scientific truth.

$$E_H(z) = -\sum_i \log P(D \mid H_i) z_i + \lambda_1 C_{\text{contradiction}} + \lambda_2 C_{\text{complexity}} - \lambda_3 C_{\text{coverage}}$$

5.6 Code generation and repair

Program synthesis naturally admits a search-and-verify structure. Candidate edits, algorithms, APIs, data structures, and compiler transformations can be encoded as choices, while tests and static analysis form a comparatively cheap verifier. This makes code one of the strongest domains for evaluating whether better candidate search translates into real task success.

6. Falsification Attempt One: Direct Neural-State Quantum Processing

6.1 Original hypothesis

The initial concept was to insert a quantum representation directly into the LLM: encode a hidden-state vector into amplitudes, apply a quantum transformation, and decode the result back into the network. The mathematical compression of indices appeared attractive because a state over n qubits spans 2^n basis states.

6.2 Discrediting attack

The proposal fails as a general architecture because the classical information must be loaded into the state and useful classical information must be recovered afterward. An asymptotic speedup inside the circuit is irrelevant if state preparation and readout scale unfavorably. Moreover, arbitrary activation transport occurs at enormous frequency inside a frontier LLM.

6.3 Verdict and amendment

Verdict: FAIL. The direct-hidden-state interface is removed from the core theory. Amendment: transfer only compact structured decision problems whose classical description is deliberately small and whose outputs are sparse candidate configurations. The amended architecture can still investigate specialized quantum machine-learning kernels later, but they are not required for Q-Deliberate to function.

7. Falsification Attempt Two: Per-Token Quantum Invocation

7.1 Original hypothesis

The second formulation assumed that any difficult local inference choice could simply be delegated to a QPU. This would place quantum calls inside the token-generation loop.

7.2 Discrediting attack

$$T_{Q,\text{total}} = T_{\text{network}} + T_{\text{queue}} + T_{\text{compile}} + T_{\text{execute}} + T_{\text{shots}} + T_{\text{measure}} + T_{\text{decode}}$$

Even if the quantum kernel itself were attractive, the total service time can greatly exceed the classical decision it replaces. Queueing and remote access are particularly damaging to tail latency. Repeated quantum calls would also cause utilization fragmentation and make cost unpredictable.

7.3 Verdict and amendment

Verdict: FAIL. Amendment: QPU work is asynchronous, coarse-grained, batched, and outside the mandatory autoregressive critical path. A classical solver always produces a deadline-safe result. A quantum result may replace it only if it arrives before the deadline and passes verification with a superior score.

$$z = z_Q \text{ if } (t_Q <= \text{deadline and } V(z_Q) > V(z_C)); \text{ else } z = z_C$$

8. Falsification Attempt Three: Quantum Superiority Assumption

8.1 Original hypothesis

After fixing data loading and latency, it is tempting to assume that combinatorial structure is sufficient to make the QPU useful. That assumption is not scientifically acceptable.

8.2 Discrediting attack

Classical optimization is extraordinarily mature. Depending on the instance, branch-and-bound, MILP, SAT/MaxSAT, simulated annealing, tabu search, local search, GPU heuristics, tensor networks, dynamic programming, Monte Carlo, graph decompositions, and problem-specific algorithms may outperform a quantum device on quality, speed, cost, or all three. A weak baseline can manufacture an apparent quantum win.

8.3 Verdict and final amendment

Verdict: the assumption FAILS. The final theory contains a compulsory quantum-value gate. Every problem family is tested against the strongest practical classical baseline under matched resource accounting. The QPU is promoted only where it establishes a statistically significant Pareto improvement. If no such region exists, QDIR remains useful as a classical deliberation framework and the quantum path is disabled.

$$U_b = Q_b - \lambda_T T_b - \lambda_C C_b - \lambda_E E_b$$

$$\text{Route to QPU only if } P(U_Q > U_C + \text{delta} \mid P) > \text{tau}$$

9. Surviving Scientific Theory

After the three failures, the surviving theory is intentionally modest and testable: there may exist structured deliberation problem distributions generated by advanced LLMs for which a quantum processor eventually produces a superior quality-cost-latency-energy frontier than classical processors, and the improvement in those subproblems may produce a measurable improvement in end-to-end LLM capability. The claim is false if either part fails.

Required condition	Failure interpretation
Relevant problem family exists	If not, quantum deliberation has no target workload.
Quantum backend improves the backend frontier	If not, use classical solvers only.
Better backend solution improves LLM task score	If not, the optimization problem is irrelevant to intelligence.
Integration overhead remains acceptable	If not, quantum speedup is operationally useless.
Result survives stronger future classical baselines	If not, prior advantage was baseline weakness.

10. Complete System Architecture

Subsystem	Primary responsibility	Failure behavior
Frontier LLM	Generate tokens, embeddings, candidate structures, and task context	Continues normal classical inference.
Candidate Generator	Create a bounded set of alternatives to optimize	Falls back to baseline decoding/search.
QDIR Compiler	Map candidates/relations/constraints into structured optimization	Reject malformed or oversized instances.
Feature Extractor	Compute size, density, graph, solver, and deadline features	Uses conservative classical route.

Subsystem	Primary responsibility	Failure behavior
Hardness/Value Router	Predict best backend and expected utility	Defaults to classical.
Classical Solver Farm	Provide strong solutions and baselines	At least one deadline-safe solver required.
Quantum Runtime	Execute selected circuits or algorithms	Optional; may time out without service failure.
Candidate Decoder	Convert states/bitstrings to structured objects	Reject impossible mappings.
Verifier	Evaluate correctness, grounding, constraints, and utility	Quantum candidate cannot pass without verification.
Telemetry/Experiment Store	Record matched runs, cost, latency, quality, energy	Missing telemetry invalidates research claim.
Distillation Engine	Teach classical model from superior quantum solutions	Disabled until superiority is established.

11. Hardness and Value Router

The router is critical because a heterogeneous computer should dispatch each computation to the physics that makes economic sense. Features should include variable count, edge density, constraint density, approximate treewidth, symmetry indicators, coefficient distributions, sparsity, known decomposition structure, historical classical solver performance, expected circuit depth, estimated QPU fidelity, queue time, shot requirement, dollar cost, energy estimate, verifier cost, and application deadline.

$$b^* = \arg\max_b U(b \mid P), \quad b \in \{CPU, GPU, QPU, FTQPU\}$$

11.1 Conservative exploration policy

Early routing should be mostly classical. A small, explicitly budgeted exploration fraction can send matched instances to quantum backends for learning. Production routing should not be driven by novelty. It should be driven by predicted utility with calibrated uncertainty.

$$P(\text{QPU route}) = \varepsilon_{\text{explore}} + (1 - \varepsilon_{\text{explore}}) I[P(\Delta U > 0) > \tau]$$

12. Quantum Algorithms and Backend Strategy

The architecture should not bet on one algorithm. The QDIR API exposes the problem and constraints; backend adapters can choose a quantum algorithm according to hardware regime. Near-term experiments may include QAOA-like variational optimization, quantum annealing where available, problem-inspired variational circuits, quantum walks on restricted graphs, or sampling procedures. Future fault-tolerant systems can investigate amplitude amplification, phase-estimation-derived subroutines, quantum walks, and algorithm-specific speedups.

12.1 QAOA-like baseline

$$|\psi(\gamma,\beta)\rangle = \prod_p e^{-i \beta_p H_M} e^{-i \gamma_p H_C} |+\rangle^{\otimes n}$$

$$H_M = \sum_i X_i$$

The project should treat QAOA as a baseline rather than a guaranteed winner. Parameter initialization, shallow depth, circuit topology, coefficient scaling, shot allocation, and noise can dominate results. QAOA performance must always be compared against exact and heuristic classical solvers on the same instances.

12.2 Avoiding trainability traps

- Prefer shallow, problem-inspired ansatzes over deep generic circuits.
- Reuse parameters between structurally similar QDIR instances.
- Grow circuit depth layer by layer only when validation improves.
- Use local or structured cost observables where possible.
- Benchmark small instances against exact state-vector simulation.

- Use tensor-network simulation to identify classically easy circuit families.
- Treat noise-aware parameter optimization and measurement calibration as part of the algorithm, not post-processing.

13. Classical Solver Tournament

A quantum claim is only credible when the classical competition is serious. Each QDIR family should maintain a solver tournament rather than a single baseline. Candidate solvers include greedy construction, beam search, simulated annealing, parallel tempering, tabu/local search, MILP, SAT/MaxSAT, CP-SAT, branch-and-bound, graph decomposition, tensor networks, GPU custom kernels, evolutionary methods, and domain-specific algorithms. The best classical result at the relevant budget becomes the comparator.

Problem characteristic	Likely classical baselines
Sparse binary quadratic	local search, simulated annealing, branch-and-bound
Dense small QUBO	exact enumeration, GPU brute force, tensor methods
Logical constraints	SAT/MaxSAT, CP-SAT
Linear/integer resource constraints	MILP, CP-SAT
Graph path/planning	A*, dynamic programming, min-cost flow, beam search
Set selection	greedy submodular methods, local improvement, MILP
Assignment/routing	Hungarian/min-cost flow, MILP, specialized heuristics

14. Verification Layer

Verification converts quantum output from an untrusted proposal into an actionable result. The ideal application exhibits search-verification asymmetry: finding a good solution is expensive, but checking one is relatively cheap. The verifier can be deterministic, probabilistic, formal, neural, or a combination, but it must be separated from the quantum solver to prevent circular evaluation.

Domain	Verifier examples
Mathematics	symbolic algebra, numerical substitution, theorem checker, unit tests
Code	compiler, test suite, fuzzing, static analysis, benchmark harness
Retrieval	citation grounding, answer entailment, source coverage, token budget
Planning	constraint checker, simulator, resource accounting, policy rules
Science	prediction-to-data comparison, consistency rules, dimensional checks
MoE routing	downstream loss, throughput, balance, capacity violations

Accept z only if Valid(z)=1 and V(z) >= V_min

15. Quantum Candidate Distribution, Not Single Answer

A QPU is naturally a sampler. Rather than using only the modal bitstring, Q-Deliberate retains a diverse set of low-energy unique states. The candidate decoder reconstructs structured plans or selections, and the verifier ranks them. Diversity itself may be the earliest useful quantum contribution, especially if quantum sampling reaches qualitatively different basins than classical heuristics.

Z = {z_1,...,z_S}; retain TopK_unique by verified objective

A near-term positive result should be described carefully. If the union of quantum and classical candidates outperforms the classical set alone, that is a hybrid utility result. It is not automatically evidence of asymptotic quantum computational advantage.

16. QDIR Compiler Training

The first compiler should be engineered manually so that researchers understand how coefficients are constructed. Once useful families are established, the compiler may become learned. A learned compiler maps LLM states, candidate graphs, and metadata into sparse optimization instances. It must be rewarded not only for downstream task quality but also for compactness, validity, and accelerator suitability.

$$\mathcal{L} = \mathcal{L}_{\text{task}} + \lambda_s \mathcal{L}_{\text{sparsity}} + \lambda_q \mathcal{L}_{\text{quantum cost}} + \lambda_v \mathcal{L}_{\text{validity}}$$

A critical risk is gaming: a learned compiler may create problems that are easy for the preferred solver but no longer faithfully represent the intended decision. Therefore, compiler fidelity tests must compare reconstructed objectives and downstream behavior against the original candidate set.

17. Quantum-to-Classical Distillation

If a QPU discovers superior decisions, those decisions can train a classical student. Over time, the student learns the easy regions of the quantum policy and QPU usage shifts toward novel hard cases. This prevents quantum resources from being wasted on solved patterns and turns the QPU into a frontier teacher rather than a permanent universal dependency.

$$S_psi(x) \sim= z_Q$$

$$f_1 = f_0 (1-d)$$

Here f_0 is the original fraction of requests using the QPU and d is the fraction successfully distilled. The architecture predicts that mature systems may use QPUs sparsely but strategically, with classical models absorbing prior quantum discoveries.

18. Experimental Program

18.1 Simulator-first ladder

1. Generate real LLM candidate problems and archive the raw candidate graph.
2. Solve small instances exactly to establish ground truth.
3. Run strong classical heuristics over matched budgets.
4. Run ideal quantum simulation to validate algorithms.
5. Run noisy simulation using hardware-calibrated error models.
6. Run the same instances on at least one physical QPU.
7. Where possible, repeat across a second hardware family.
8. Feed solutions back to a frozen LLM and measure end-to-end task effects.

18.2 Benchmark families

Family	Example task	Primary measurement
Reasoning	math/science multi-step problems	verified answer accuracy per compute budget
Code repair	repository-level bug fixing	tests passed, wall time, cost
Retrieval	long-context question answering	grounded answer quality vs context tokens
Planning	tool-use workflows	success rate under action/cost limits
Scientific synthesis	model/hypothesis combination	held-out predictive fit and consistency
MoE routing	blockwise expert selection	loss/quality vs throughput and balance
Synthetic CSP/QUBO	controlled random ensembles	scaling law and optimality gap

19. Statistical Design and Falsification Criteria

Experiments should use paired instances, preregistered metrics, and confidence intervals. For task i, define d_i as the difference between the QPU-assisted and best classical conditions. The null hypothesis should be that the quantum condition provides no positive improvement.

$$H0: E[d_i] <= 0; \quad H1: E[d_i] > 0$$

Bootstrapping, permutation testing, or hierarchical modeling can be used depending on benchmark structure. Multiple-comparison correction is required if many problem families are tested. Reporting only the best-performing subset after the fact is not acceptable.

19.1 Mandatory ablations

- QDIR with QPU disabled.
- QDIR with random sampling instead of optimization.
- QDIR with greedy solver.
- QDIR with best tuned classical solver.
- Quantum solver with verifier disabled, for diagnosis only.
- Router disabled with forced classical and forced quantum routes.
- Compiler replaced by random or simplified formulations.
- Quantum candidates added only for diversity versus used for objective improvement.

20. Success Metrics

20.1 Backend-level metrics

Metric	Definition
Optimality gap	difference from exact or best-known objective
Time-to-quality	wall time required to cross a solution-quality threshold
Dollar-to-quality	provider + compute cost required for target quality
Energy-to-quality	estimated or measured joules for target quality
Diversity	distinct high-quality solution basins returned
Reliability	fraction of jobs producing valid solutions before deadline

20.2 End-to-end LLM metrics

$$\Delta S = S_Q - S_{\text{base}}$$

The strongest claim is an increase in task score at approximately equal total resource budget. Alternative valid wins include equal score at lower cost, lower latency, or lower energy. A backend win without model-level benefit does not satisfy the primary objective.

21. Scaling-Law Analysis

Rather than focusing on one finite instance, the program should estimate empirical scaling curves. Let T_C(n) and T_Q(n) be time-to-quality for a classical and quantum solver as instance size increases. A quantum processor can have worse constant factors at small n yet become relevant if its growth exponent is more favorable.

$$T_C(n) \approx a_C e^{\alpha n}, \quad T_Q(n) \approx a_Q e^{\beta n}$$

$$n^* = \ln(a_Q/a_C) / (\alpha - \beta), \text{ if } \beta < \alpha$$

This model is illustrative rather than universal. Power-law, subexponential, or piecewise fits may be more appropriate. The key principle is to estimate a crossover region instead of declaring success from isolated examples.

22. Shadow-Mode Deployment

The first production-like deployment should run quantum decisions in shadow mode. Classical output controls the user-visible LLM. The quantum path operates simultaneously, records what it would have selected, and is scored by the verifier and downstream task. This produces paired operational data without risking service quality.

Promotion criteria should be fixed in advance, for example: minimum sample count, positive lower confidence bound on model-level improvement, no violation of safety or reliability limits, and acceptable cost/latency envelope. A single impressive demonstration is insufficient.

23. Minimum Viable Experiment

A practical first experiment can be conducted without training a new foundation model. Use a capable open-weight model or a frontier model API. Choose long-context retrieval or constrained planning. Ask the LLM to generate 30-80 candidate items, prune to a 20-80 variable QDIR instance, solve with exact methods where possible, strong GPU/classical heuristics, ideal QAOA simulation, noisy simulation, and one physical QPU. Feed each solver result back into the identical frozen LLM and compare verified task outcomes.

Element	MVE specification
LLM	frozen model; identical weights across conditions
Candidate count	30-80 before pruning
QDIR variables	20-80 initial target
Classical baselines	exact when feasible + at least 3 heuristic families
Quantum	simulator + physical QPU
Primary endpoint	paired verified task score
Secondary endpoints	latency, cost, energy proxy, optimality gap, diversity
Replication	multiple random seeds and repeated physical runs

24. Hardware and Infrastructure Specification

The early project should not purchase or fabricate a dedicated quantum computer. Quantum access should be treated as an external accelerator service until utilization and economic advantage justify tighter integration. The main capital requirement is conventional AI/HPC infrastructure for LLM inference, quantum simulation, exact solving, telemetry, and data analysis.

Resource	Lean prototype	Serious research cluster
AI accelerators	8-32 high-end GPUs	32-128 high-end GPUs
CPU nodes	4-8	8-32
Aggregate RAM	2-8 TB	8-64 TB
Fast storage	100-500 TB	0.5-2 PB
Network	100-200 Gb/s	200-800 Gb/s
Quantum access	1 provider	2+ providers / hardware families
Quantum simulation	state-vector on GPU	state-vector + tensor-network + noise simulation
Scheduler	Ray/Slurm/Kubernetes	integrated HPC + service scheduler
Experiment store	PostgreSQL + object store	warehouse + object store + lineage system

25. Software Architecture and APIs

Layer	Recommended implementation
Model runtime	PyTorch/JAX/vendor API
Candidate generation	model-specific service with structured output
QDIR compiler	Python prototype, C++/Rust optimization later
Serialization	Protocol Buffers or equivalent typed schema
Classical optimization	OR-Tools/CP-SAT, MILP, custom GPU solvers, SAT/MaxSAT
Quantum abstraction	CUDA-Q and/or Qiskit adapters
Quantum simulation	GPU state-vector and tensor-network backends
Service transport	gRPC with authenticated TLS
Scheduling	Kubernetes + Slurm/Ray as appropriate
Telemetry	OpenTelemetry/Prometheus-like stack
Experiment tracking	versioned artifact store and immutable run manifests
Verification	domain-specific sandboxed services

25.1 Reference request flow

Context -> LLM candidates -> QDIR -> Router -> parallel solvers -> candidate pool -> verifier -> LLM continuation

The system must preserve deterministic identifiers for every stage so researchers can reconstruct exactly which model, prompt template, compiler version, coefficient normalization, solver settings, quantum calibration, and verifier version produced an output.

26. Data Model and Telemetry

Every solver call should create a durable experiment record. Minimum fields include problem hash, raw candidate provenance, QDIR family, variable count, coefficients, constraints, compiler version, backend, hardware identifier, calibration snapshot if available, circuit identifier, depth, gate counts, shot count, queue delay, execution time, post-processing time, financial cost, energy estimate, candidate states, decoded solutions, verifier scores, and final LLM task outcome.

Missing telemetry should invalidate any comparative claim because interface overhead is part of the system. Quantum kernel time alone is not an acceptable performance denominator.

27. Security, Privacy, and Trust Boundaries

The QPU gateway should normally receive QDIR rather than raw user prompts or private model chain-of-thought text. QDIR is numerical and structured, reducing unnecessary information exposure. Transport must be encrypted and authenticated. Vendor-visible metadata should be minimized. Returned quantum results are untrusted computation until decoded and verified inside the trusted environment.

- Never allow QPU output to bypass model safety or application policy checks.
- Do not expose raw sensitive prompts when an abstract QDIR representation is sufficient.
- Sandbox code and tool-plan verification.
- Hash and sign experiment manifests for scientific reproducibility.
- Use least-privilege credentials for each external QPU provider.
- Retain provider-specific data according to organizational privacy policy.

28. Reliability and Failure Containment

Quantum computation must be optional at runtime. The architecture therefore uses a deadline-safe classical fallback. QPU outages, calibration changes, queue spikes, invalid samples, or software failures reduce experimental opportunity but do not block ordinary model service.

Failure	Required response
QPU unavailable	route to best classical solver
Queue exceeds deadline	cancel/ignore QPU job; use classical result
Invalid decoded state	reject candidate
Verifier disagreement	retain classical/default result or escalate according to application
Quantum quality degradation	router lowers predicted utility for that hardware/calibration
Telemetry unavailable	disable research promotion claims for affected run
Provider API change	adapter isolation prevents model-runtime failure

29. Personnel and Organization

A lean 12-18 month prototype can be built by roughly 8-15 highly capable researchers and engineers. A mature 35-60 person program is more appropriate for multi-domain benchmarks, multi-QPU support, security, production shadow deployment, and a fault-tolerant research track.

Role	Lean FTE	Mature FTE	Responsibilities
Quantum algorithms	2-3	6-10	algorithms, circuits, resource analysis
LLM research	2-3	6-10	candidate generation, reasoning, benchmarks
Optimization/classical baselines	1-2	4-6	solver tournament, hardness analysis
ML/HPC systems	2-3	6-10	GPU stack, scheduler, services

Role	Lean FTE	Mature FTE	Responsibilities
Compiler/runtime	1-2	4-6	QDIR, backend adapters, caching
Evaluation/statistics	1	3-5	preregistration, analysis, scaling laws
Security/operations	shared	3-5	trust boundaries, deployment, reliability
Program leadership	1	2-4	technical direction, partnerships, gates

30. Project Schedule and Stage Gates

Stage	Calendar	Core deliverable	Go/no-go gate
0 - theory validation	0-3 months	QDIR taxonomy, real workload corpus, classical hardness map	stop quantum track if no useful hard problem distributions emerge
I - simulator prototype	3-9 months	QDIR v1, solver tournament, ideal/noisy simulation	drop algorithm families that scale worse than classical alternatives
II - physical QPU	9-18 months	matched multi-backend experiments	continue only families with credible scaling or complementary utility
III - frontier LLM shadow mode	18-30 months	asynchronous runtime, learned router, paired production-like data	stop families where solver improvement does not improve LLM outcomes
IV - quantum-value demonstration	24-42 months	statistically defensible Pareto region, if one exists	scale only after preregistered success threshold
V - fault-tolerant transition	hardware triggered	deep algorithms on logical qubits	proceed only after resource estimates and actual hardware support it

31. Cost Model

The following ranges are engineering planning estimates rather than current vendor bids. Costs depend strongly on geography, compensation, GPU ownership versus cloud usage, QPU partnerships, benchmark scale, and whether the organization already operates an AI cluster.

Category	Three-year planning range
Research personnel	US$8M-18M
AI/GPU infrastructure	US$4M-12M
Quantum access and partnerships	US$2M-8M
Storage/networking	US$1M-4M
Software/infrastructure	US$2M-5M
Benchmark/evaluation	US$1M-3M
Security/operations	US$1M-3M
Contingency	US$3M-8M
Indicative total	US$22M-61M

31.1 Funding tiers

Tier	Budget	What it can credibly answer
Exploratory	US$0.4M-1M	Do real LLM workloads yield meaningful QDIR structures?
Minimum rigorous prototype	US$2M-5M	Can simulator + physical QPU experiments be run against strong baselines?
Integrated program	US$12M-35M cumulative	Can Q-Deliberate be attached to a high-end LLM in shadow mode?
Three-year serious program	US$22M-61M	Can multiple domains/backends establish a reproducible quantum-value region?
Industrial FTQC transition	US$50M-150M+	Can proven QDIR families migrate to fault-tolerant algorithms?

These estimates exclude training a new frontier LLM from scratch and exclude constructing a quantum computer. The economically rational early strategy is to attach Q-Deliberate to an existing LLM and rent or partner for quantum access.

32. Major Technical Risks

Risk	Why it matters	Mitigation
No relevant quantum advantage	classical solvers may dominate all workloads	make classical-only QDIR useful; stop quantum spending at gates
Data loading returns indirectly	compiler may create overly large dense instances	strict representation budget and sparsity targets
QPU latency	can destroy interactive utility	asynchronous batching, deadlines, shadow mode
Noise/trainability	variational circuits may not scale	shallow structured circuits, simulation, multiple algorithms
Weak classical baselines	false-positive quantum claims	solver tournament and continuously refreshed baselines
Benchmark gaming	compiler may optimize metrics rather than task	held-out benchmarks, frozen model comparisons, ablations
No causal LLM benefit	better QUBO score may not improve intelligence	measure dS/dQ_D and terminate irrelevant families
Vendor dependence	hardware roadmap may slip	backend-neutral QDIR and multi-provider adapters
Cost explosion	QPU and GPU experiments can scale rapidly	hard per-stage budgets and sample-size planning

33. Fault-Tolerant Extension

The long-term architecture separates NISQ-oriented and fault-tolerant lowering. QDIR_NISQ prioritizes small sparse instances, shallow circuits, topology compatibility, and robustness to noise. QDIR_FT prioritizes algorithmic complexity and logical-qubit efficiency. The same high-level decision problem may compile very differently in the two regimes.

$$QDIR \rightarrow \{Lower\_NISQ(P,H), Lower\_FT(P,H)\}$$

Fault-tolerant resource estimation should include logical qubits, T-count/T-depth or analogous costly logical operations, code distance, physical qubits per logical qubit, factory requirements, runtime, and total error budget. Project milestones must be tied to demonstrated hardware capability rather than promised calendar dates.

34. Recursive Quantum Deliberation

If a single quantum deliberation step proves useful, a future architecture can iterate. The LLM formulates P_1, receives candidate z_1, interprets it, formulates P_2, and so on. The resulting system alternates neural generation with structured physical search. This should not be attempted until one-shot deliberation is validated because recursion multiplies latency, evaluation complexity, and attribution difficulty.

$$x \rightarrow P_1 \rightarrow z_1 \rightarrow P_2 \rightarrow z_2 \rightarrow \ldots \rightarrow y$$

35. Deliberation Kernel Cache

Repeated QDIR structures can be cached. A structural signature can index previously successful circuit layouts, variational parameters, classical heuristics, embedding choices, decomposition strategies, and hardware-specific performance. The purpose is to amortize circuit compilation and parameter search over recurrent problem families.

$$Cache[sig(P)] = \{layout, params, solver\ priors, performance\ model\}$$

The cache should be invalidated or down-weighted when hardware calibration, compiler version, coefficient distribution, or problem-family statistics change materially.

36. Long-Range Implication: Heterogeneous Reasoning Machines

The deeper claim is architectural rather than algorithm-specific. Advanced AI systems may increasingly resemble heterogeneous computers in which neural inference, symbolic verification, combinatorial optimization, database

retrieval, simulation, and eventually quantum algorithms are dispatched to different computational substrates. The LLM becomes a problem-formulation and orchestration layer, not the sole computational mechanism.

Neural inference + structured search + verification + distillation = heterogeneous reasoning system

If quantum processors never obtain a useful region on these workloads, the architecture collapses gracefully into a classical deliberation fabric. If they do, the interface is already present. This asymmetry makes the research program comparatively robust to uncertainty in quantum hardware progress.

Appendix A. Reference QDIR Schema

The following schema is illustrative and should be implemented in a typed serialization format. Coefficients should be numerically normalized and accompanied by explicit units or semantic meaning where applicable.

```
QDIRProblem {
  problem_id: UUID
  family: enum
  compiler_version: string
  model_version: string
  variable_count: integer
  variable_metadata: repeated Variable
  linear_terms: repeated (i, coefficient)
  quadratic_terms: repeated (i, j, coefficient)
  hard_constraints: repeated Constraint
  soft_constraints: repeated Constraint
  decoder_spec: DecoderReference
  verifier_spec: VerifierReference
  deadline_ms: integer
  max_cost_usd: float
  max_energy_j: float or null
  minimum_confidence: float
  provenance_hash: bytes
}
```

Appendix B. Reference Orchestration Pseudocode

```
function Q_DELIBERATE(context):
    candidates = LLM.generate_candidates(context)
    problem = QDIR.compile(candidates)
    features = analyze(problem)
    route = router.predict(features)
    c_future = classical_solver.solve_async(problem)
    q_future = null
    if route.allows_quantum:
        q_future = quantum_solver.solve_async(problem)
```

```
c_result = c_future.result_by(deadline)

best = verifier.rank(c_result)

if q_future and q_future.finished_by(deadline):

    q_result = q_future.result()

    q_verified = verifier.rank(q_result)

    if q_verified.score > best.score:

        best = q_verified

log_all(problem, c_result, q_result, best)

return LLM.continue_with(best)
```

Appendix C. Reference Acceptance Gates

Gate	Minimum evidence before passing
G0 - workload relevance	real LLM-generated structured problems exist and influence task outcomes
G1 - compiler fidelity	QDIR solution quality correlates with intended decision quality
G2 - classical challenge	strong baseline tournament implemented and tuned
G3 - simulator correctness	ideal quantum solver matches known small-instance behavior
G4 - physical reproducibility	results replicate across repeated hardware runs
G5 - backend utility	quantum/hybrid route beats classical frontier on preregistered metric
G6 - end-to-end utility	LLM task score/cost/latency/energy improves under matched budget
G7 - operational safety	shadow deployment shows no unacceptable reliability or security regression
G8 - scale decision	projected crossover remains credible under improved classical baselines

Appendix D. Example Retrieval QUBO Construction

Suppose the retriever proposes N chunks. Each chunk i has relevance r_i, token length l_i, source identity s_i, and embedding. Let d_ij be redundancy similarity and g_ij be a learned synergy score estimating whether the two chunks jointly support an answer. A binary variable z_i selects chunk i.

$$E(z) = -\sum_i r_i z_i + \lambda_d \sum_{i<j} d_{ij} z_i z_j - \lambda_g \sum_{i<j} g_{ij} z_i z_j + \lambda_B [\max(0, \sum_i \ell_i z_i - B)]^2$$

Additional penalties can discourage selecting too many chunks from a single source, reward temporal diversity, or enforce mandatory inclusion of authoritative evidence. The exact same QDIR instance is solved by classical and quantum backends. The final comparison occurs only after the selected chunks are fed to the same frozen LLM and the answer is scored for correctness and grounding.

Appendix E. Example Planning QUBO

For a set of candidate actions a_i and candidate precedence edges e_ij, variables may represent action selection and ordering. The objective rewards expected task utility and penalizes cost, incompatible action pairs, missing prerequisites, and illegal cycles. Because general cycle constraints can be expensive in a simple QUBO, the compiler may use a layered time-step representation or a backend that supports richer constraints natively.

$$E_{\text{plan}} = -\sum_i u_i z_i + \lambda_c \sum_i \text{cost}_i z_i + \lambda_x \sum_{i<j} \text{conflict}_{ij} z_i z_j + \lambda_p P_{\text{prereq}} + \lambda_o P_{\text{order}}$$

Appendix F. Measurement and Error Accounting

For noisy physical circuits, raw kernel time is not enough. The experiment must count compilation, queueing, shot repetitions, readout calibration, post-processing, and discarded invalid samples. A crude independent-error expression can be useful for intuition but should not replace hardware-specific characterization.

$$F_{\text{approx}} = (1 - p_1)^{N_1}(1 - p_2)^{N_2}(1 - p_m)^{N_m}$$

Reported results should include gate counts, two-qubit depth, measurement count, shot count, calibration timestamp, qubit mapping, and transpiler/circuit compiler version. If error mitigation is used, both mitigated and unmitigated results should be archived.

Appendix G. Project Deliverables

Milestone	Deliverable
M1	QDIR v0.1 specification and 10,000-instance workload corpus
M2	Classical solver tournament with exact small-instance ground truth
M3	Ideal and noisy quantum simulation harness
M4	First physical-QPU matched benchmark report
M5	QDIR compiler SDK and backend-neutral runtime
M6	Hardness/Value Router v1 with calibrated utility estimates
M7	Shadow-mode frontier LLM integration
M8	Preregistered end-to-end quantum-value study
M9	Distillation prototype, conditional on superior quantum candidates
M10	Fault-tolerant resource-estimation report for winning families

Appendix H. Reference Risk Register

ID	Risk	Probability	Impact	Trigger	Response
R1	No quantum-beneficial problem family	High	High	18-month data show no favorable scaling	stop physical-QPU expansion; continue classical QDIR only
R2	Quantum hardware roadmap delay	High	Medium	logical-qubit milestones slip	keep milestones hardware-triggered; use simulation
R3	Classical solver leapfrogs QPU	High	Medium	new algorithm dominates prior advantage	rebenchmark and withdraw quantum route if needed
R4	Integration latency too large	Medium	High	QPU misses deadlines	batch, prefetch, shadow mode, offline use
R5	Learned compiler gaming	Medium	High	objective improves but task score drops	fidelity tests and held-out evaluations
R6	Cost exceeds benefit	Medium	High	dollar-to-quality worsens	router disables expensive quantum path
R7	Security/privacy exposure	Low-Med	High	sensitive data reaches external provider	QDIR abstraction, redaction, policy enforcement

Appendix I. Selected Background Sources

1. NVIDIA CUDA-Q documentation and developer material: heterogeneous CPU/GPU/QPU programming and quantum simulation/runtime concepts. https://developer.nvidia.com/cuda-q

2. IBM Quantum Roadmap: current and planned quantum-centric supercomputing and fault-tolerant milestones. https://www.ibm.com/roadmaps/quantum/

3. IBM Quantum processor documentation: processor families and hardware specifications. https://quantum.cloud.ibm.com/docs/en/guides/processor-types

4. Google Quantum AI / Willow overview: error-correction and quantum-computing roadmap context. https://blog.google/innovation-and-ai/technology/research/google-willow-quantum-chip/

5. Microsoft Azure Quantum Resource Estimator: fault-tolerant resource-estimation methodology. https://learn.microsoft.com/en-us/azure/quantum/overview-resources-estimator

6. General research literature on barren plateaus, quantum optimization, quantum machine learning data loading, QAOA, tensor-network simulation, and fault-tolerant resource estimation should be incorporated into any peer-reviewed continuation of this project.

Appendix J. Scientific Status Statement

Q-Deliberate is a research theory and systems proposal. It does not establish that present quantum computers improve frontier LLMs, nor does it claim a proven asymptotic speedup for the proposed workloads. The principal contribution is an architecture and experimental methodology that makes the question falsifiable while avoiding three major failure modes: dense classical data loading, token-level quantum latency, and weak classical comparison. Any future claim of advantage should be expressed in terms of matched end-to-end quality, cost, latency, and energy measurements with complete experimental provenance.