Quantum–Enhanced NEAT: A Hybrid Quantum–Classical Neuroevolution Framework

Author: Nicholas Starbuck (RedFox-AI51)

Affiliation: Independent Researcher, NSW, Australia

Date: 26 August 2025

Abstract

NeuroEvolution of Augmenting Topologies (NEAT) efficiently evolves both weights and topology of neural networks via speciation, historical markings, and incremental complexification. Real-Time NEAT (rtNEAT) further enables online replacement in interactive settings. This paper proposes **Quantum–Enhanced NEAT (QE-NEAT)**: a practical hybrid quantum–classical framework that augments NEAT/rtNEAT with (i) **quantum-inspired operators** that improve search diversity and credit assignment on classical hardware, and (ii) **variational quantum subroutines** that accelerate discrete choices (e.g., mutation selection, parent selection, and module routing) via quadratic unconstrained binary optimization (QUBO) mappings or amplitude-amplification–like samplers when suitable hardware is available. QE-NEAT targets **spiking neural networks (SNNs)** and conventional ANNs, with an emphasis on **event-driven tasks** and **online control**. We outline the system architecture, genome encodings, speciation-compatible quantum operators, and an evaluation plan across temporal benchmarks (e.g., DVS Gesture, N-MNIST), control (CartPole-v1, MuJoCo), and online-learning scenarios.

Keywords: NEAT, rtNEAT, quantum-inspired evolutionary algorithms, variational quantum algorithms, QAOA, spiking neural networks, neuromorphic computing, hybrid quantum–classical optimization

---

1. Introduction

Neuroevolution explores parameter and topology spaces using evolutionary operators, providing a robust alternative to gradient-based learning—particularly attractive when gradients are unavailable, deceptive, or expensive. NEAT's hallmark contributions—**historical markings** for safe crossover, **speciation** for protecting innovation, and **complexification** from minimal genomes—established a strong baseline for discrete structure search. rtNEAT further demonstrated real-time replacement in interactive environments.

Problem. As network complexity and task difficulty grow—especially for SNNs and closed-loop control—evolutionary search may suffer from premature convergence, weak credit assignment across interacting modules, and high wall-clock cost.

Proposal. QE-NEAT augments NEAT/rtNEAT with: - **Quantum-inspired operators (QIOs):** Classical algorithms that borrow state superposition–style diversity and rotation-gate–style update rules to balance

exploration/exploitation without quantum hardware. - **Variational quantum subroutines (VQSs):** Optional modules that map discrete subproblems (e.g., module selection, parent pairing, routing sparsification) to QUBO/Ising forms and solve them with shallow-depth variational circuits (e.g., QAOA/VQE-style) or specialized samplers when quantum access exists.

We focus on **spiking policies** (event-driven inference, temporal coding) and **real-time evolution** in the loop, targeting improved sample-efficiency, search diversity, and stability.

Contributions. 1. A cohesive **system architecture** for hybrid quantum–classical evolutionary search integrated with NEAT/rtNEAT and SNNs. 2. **Genome encodings** and **operators** that are speciation-compatible and admit QUBO mappings. 3. A **methodology** for fair comparison to strong classical baselines with transparent ablations. 4. An **engineering blueprint** for a solo developer to implement and iterate pragmatically.

---

2. Background

2.1 NEAT and rtNEAT

- **Historical markings** (innovation numbers) to align genomes during crossover.
- **Speciation** via distance metric $\delta = c_1 \frac{E}{N} + c_2 \frac{D}{N} + c_3 \bar{w}$ to protect structural innovation.
- **Complexification** from minimal structure using add-node/add-connection mutations.
- **rtNEAT**: steady-state replacement; species quotas and real-time fitness updates enable online adaptation.

2.2 Spiking Neural Networks (SNNs)

SNNs model neuron dynamics (e.g., LIF) and communicate via spikes; they are event-driven and potentially energy-efficient. Learning rules include STDP variants, surrogate-gradient training, and neuroevolution. Evolution is appealing when differentiability or stable surrogate gradients are hard to obtain.

2.3 Quantum(-inspired) Optimization

- **Quantum-inspired evolutionary algorithms (QIEA):** Classical EAs using qubit-like probability amplitudes and rotation updates to maintain diverse populations.
- **Variational quantum algorithms (VQAs):** Hybrid quantum–classical training of parametrized circuits; relevant example: **QAOA** for discrete optimization.

---

3. Related Work

- **NEAT / rtNEAT**: Topology+weight evolution and real-time replacement.
- **Neuroevolution for SNNs**: Encoding synaptic connectivity/topology and neuron parameters; indirect encodings (e.g., CPPNs) scale to larger SNNs.
- **Quantum-inspired EAs**: Demonstrated improvements in exploration and convergence in classical settings.

- **Quantum neuroevolution**: Markovian quantum neuroevolution (MQNE) for quantum circuit architecture search shows that evolutionary search can effectively optimize discrete circuit structures.
- **Quantum spiking models**: Emerging proposals for quantum spiking neurons and stochastic quantum SNNs hint at hybrid neuromorphic–quantum opportunities.

---

4. QE-NEAT Architecture

4.1 High-Level Loop

1. **Population init** (minimal genomes; SNN or ANN nodes).
2. **Fitness evaluation** (episodic or online, possibly with spike metrics and energy proxies).
3. **Speciation & quotas** using NEAT distance; maintain target species count.
4. **Parent selection** per species (fitness sharing).
5. **Variation:**
6. **Crossover** aligned by innovation numbers.
7. **Mutations** (add-node, add-connection, enable/disable) plus **SNN-specific**: threshold, time-constant, synaptic delay, STDP hyperparameters.
8. **Routing/Sparsification**: optional QUBO/VQA step to select a near-optimal sparse subgraph/module composition under constraints.
9. **Replacement** (steady-state). For **rtNEAT**, inject offspring in real time; remove worst individuals species-wise.

4.2 Genome Encoding for SNNs

- **Node genes:** neuron type (LIF, adaptive LIF), parameters ($\tau_m, \tau_{ref}, V_{th}$), optional adaptation terms.
- **Connection genes:** pre/post IDs, weight, delay, plasticity flag, STDP rule ID + hyperparameters.
- **Module genes (optional):** reusable microcircuits (e.g., convolutional SNN blocks) with interface pins. Modules allow coarse-grained search and QUBO-friendly selection.

4.3 Quantum–Enhanced Operators

A. Quantum-Inspired Mutation Scheduler (QIMS): - Maintain a vector of mutation operators with probability amplitudes $\mathbf{q}$ (complex or 2D real proxy). After each generation, update via rotation-like rules toward operators with higher marginal fitness contributions; enforce lower bounds to prevent collapse.

B. Variational QUBO for Routing/Sparsification (VQRS): - Map selection of edges/modules under budget B to $\min_{x\in\{0,1\}^n} x^T Q x$ with constraints via penalties. - Solve via: (i) classical QUBO solvers, (ii) simulated QAOA (parameter-shift gradients), or (iii) hardware QAOA when accessible.

C. Quantum-Inspired Parent Selection (QIPS): - Maintain a **mixture sampler** over species-lineage “beams”. Use amplitude-scaling updates to emphasize recently successful lineages while retaining long-tail exploration.

D. Quantum-Inspired Innovation Reuse (QIIR): - Treat innovation numbers as a dictionary; periodically sample “innovation superpositions” that propose reusing historical modules weighted by context similarity (task phase, sensory stats).

4.4 rtNEAT Integration

- Keep **insertion rate** and **lifetime** constraints per species.
- Use **VQRS** opportunistically at low frequency (e.g., every K seconds or after score plateaus) to avoid real-time stalls.
- Maintain **hot-swappable** SNN modules; when a module is replaced, reload last good state if applicable.

---

5. Algorithmic Details

5.1 Distance Metric (Speciation)

$$\delta(\mathbf{g}_1, \mathbf{g}_2) = c_1 \frac{E}{N} + c_2 \frac{D}{N} + c_3 \bar{w} + c_4 \bar{\Delta d} + c_5 \bar{\Delta \tau}$$

Add SNN-specific terms: average synaptic delay difference $\bar{\Delta d}$ and time-constant difference $\bar{\Delta \tau}$.

5.2 Quantum-Inspired Mutation Update (per generation)

1. Compute marginal contribution of each operator via fitness attribution (e.g., Shapley-style approximation or ablation counts).
2. Update operator amplitudes $q_j \leftarrow R(\alpha_j, \beta_j) q_j$ with rotation parameters set by normalized contributions and diversity pressure.
3. Sample actual mutation set from $|q_j|^2$ after applying floors/ceilings.

5.3 VQRS Mapping Sketch

- Variables: x_e for candidate edges/modules.
- Objective: $Q = \lambda_1 \cdot \text{complexity}(x) - \lambda_2 \cdot \text{utility}(x) + \lambda_3 \cdot \text{constraint penalties}$.
- Utility proxies: information gain of connections (mutual information between pre/post spike trains), contribution to policy value, or temporal-coverage scores.
- Solve with classical QUBO first; optionally cross-check with simulated QAOA. Cache solutions per species to amortize cost.

5.4 SNN Training Options per Individual

- **Pure evolution** (weights + topology + neuron params).
- **Memristive/plastic synapses enabled**: evolution sets STDP meta-parameters while runtime plasticity adapts online.
- **Hybrid fine-tuning**: brief surrogate-gradient steps (few iterations) after evolutionary placement to polish weights.

---

6. Experimental Plan

6.1 Benchmarks

- **Event-based vision:** N-MNIST, DVS Gesture.
- **Control (classic):** CartPole-v1, Acrobot, MountainCar.
- **Control (continuous):** MuJoCo HalfCheetah/Ant with spike-encoded observations.
- **Online/interactive:** simple arena (NERO-like) with curriculum.

6.2 Baselines & Ablations

- **Baselines:** NEAT (classical), rtNEAT, CMA-ES topo-evolution (if applicable), surrogate-gradient SNN.
- **Ablations:** (i) remove QIMS, (ii) remove VQRS, (iii) remove both (plain NEAT), (iv) classical QUBO vs QAOA backends, (v) no SNN-specific terms in speciation.

6.3 Metrics

- **Sample-efficiency:** reward vs. environment steps.
- **Wall-clock:** time-to-threshold; eval throughput.
- **Search quality:** best/median fitness; species diversity; innovation survival curves.
- **Model cost:** parameter count, synapse count, average firing rate, energy proxy (spike events/sec).
- **Stability:** variance across seeds; catastrophic forgetting in online settings.

6.4 Protocol

- Fixed seeds, 10–20 independent runs per setting.
- Resource budget caps (episodes, time, qubits if used).
- Pre-register hyperparameters; tune on a small subset and lock before full runs.
- Report confidence intervals; use nonparametric tests where appropriate.

---

7. Engineering Blueprint (Solo-Developer Focus)

Runtime: - **Core loop:** Python (Rust/C++ where hotspots appear). Multiprocessing for evaluation workers. - **Numerics:** NumPy/JAX for classical ops; PyTorch for SNN simulation (e.g., Norse, SpikingJelly) or custom CUDA kernels for LIF. - **QUBO:** PyQUBO or custom; classical solvers (qbsolv, simulated annealing). Optional: Qiskit/PennyLane for simulated QAOA. Abstract an **OptimizerBackend** interface. - **Genome store:** append-only log of innovations; mmap-backed arrays for fast species slicing. - **Telemetry:** Prometheus exporters for population stats, species sizes, innovation entropy, eval throughput.

Data Structures: - **Struct-of-arrays (SoA)** for genomes to enable vectorized distance metrics. - **Ring buffers** for spike trains; compressed sparse connectivity for SNNs.

Performance Tactics: - JIT critical kernels (JAX/XLA or Numba). Batch distance computations. Cache speciation buckets. - Stagger VQRS invocations (e.g., every K generations or on plateau) to control overhead.

Reproducibility: - Deterministic RNG streams per operator. Versioned innovation table. Experiment manifests (YAML). Auto-archived seeds, code commit, and env hashes.

---

8. Limitations, Risks, and Mitigations

- **NISQ fragility & overhead:** Treat quantum backends as *optional accelerators*. Always provide classical fallbacks and report parity.
- **Barren plateaus / optimizer pathologies:** Prefer shallow ansätze, layerwise training, or warm-starts. Resort to gradient-free QAOA parameter scans when gradients fail.
- **Attribution noise in QIMS:** Use moving windows and robust attribution (e.g., bootstrapped contributions) to avoid oscillatory probability collapse.
- **Scaling SNN genomes:** Use module genes and indirect encodings (e.g., CPPN) to control search dimensionality.

---

9. Ethical and Safety Considerations

- **Compute/energy:** Track and report energy proxies; prefer event-driven inference and sparsity.
- **Dual-use:** Restrict high-speed autonomous control experiments to simulators; clearly disclose constraints if/when transferring to hardware.
- **Reproducibility:** Release code, configs, seeds, and result artifacts.

---

10. Milestones & Timeline

1. **M0–M1:** Minimal NEAT+SNN engine; genome & speciation with SNN terms.
2. **M1–M2:** QIMS operator; ablation vs. NEAT.
3. **M2–M3:** VQRS QUBO (classical); ablation & plateau-triggered runs.
4. **M3–M4:** Optional QAOA backend (simulator); parity study.
5. **M4–M5:** rtNEAT online loop; interactive arena demo.
6. **M5–M6:** Full benchmarks, reporting, artifact release.

---

11. Conclusion

QE-NEAT integrates quantum-inspired diversity management and optional variational quantum subroutines with the proven NEAT/rtNEAT machinery to improve search quality and practicality for spiking and standard neural networks—particularly in temporal and online-control tasks. The framework is designed to be **hardware-agnostic**, enabling immediate classical evaluation while remaining ready for opportunistic quantum acceleration.

References (selected)

[1] Stanley, K. O., & Miikkulainen, R. *Evolving Neural Networks through Augmenting Topologies*. Evolutionary Computation, 10(2), 99–127, 2002. [2] Stanley, K. O., Bryant, B. D., & Miikkulainen, R. *Real-Time Neuroevolution in the NERO Video Game*. IEEE Trans. Evolutionary Computation, 9(6), 653–668, 2005. [3] Cerezo, M., et al. *Variational Quantum Algorithms*, arXiv:2012.09265, 2020. [4] Farhi, E., et al. *A Quantum Approximate Optimization Algorithm*, arXiv:1411.4028, 2014. [5] Han, B., et al. *Learning Rules in Spiking Neural Networks: A Survey*. Neurocomputing, 2023. [6] Narayanan, A., et al. *Quantum-Inspired Evolutionary Algorithms: A Survey and Empirical Study*. J. of Heuristics, 2011. [7] Lu, Z., Shen, P.-X., & Deng, D.-L. *Markovian Quantum Neuroevolution for Machine Learning*. Phys. Rev. Applied 16, 044039 (2021). [8] Kristensen, L. B., et al. *An Artificial Spiking Quantum Neuron*. npj Quantum Information, 2021. [9] ORNL/UCF. *Neuroevolution of Spiking Neural Networks using CPPNs*. Research highlight, 2021. [10] Bravo-Prieto, C., et al. *Variational Quantum Linear Solver*. arXiv:1909.05820, 2019. [11] Wierichs, D., et al. *General Parameter-Shift Rules for Quantum Gradients*. Quantum 6, 677 (2022). [12] Tan, C., Šarlija, M., & Kasabov, N. *Spiking Neural Networks: Background, Recent Development and the NeuCube Framework*. Neural Computing and Applications, 2020.