

qanneal: A CPU-First Simulated Quantum Annealing Toolkit

qanneal contributors

February 21, 2026

Abstract

This report describes the mathematical foundations and software design of **qanneal**, a CPU-first simulated quantum annealing (SQA) toolkit. We detail the Ising and QUBO formulations, the Suzuki–Trotter mapping for transverse-field models, and the Monte Carlo algorithms used in classical SA and SQA. We also document the code architecture, trace observability, and practical complexity considerations.

1 Introduction

Combinatorial optimization problems can often be written as minimizing a quadratic energy function over binary variables. Simulated annealing (SA) provides a classical stochastic method for approximately solving these problems. Simulated quantum annealing (SQA) extends this approach by emulating quantum fluctuations through a path-integral representation of a transverse-field Ising model. **qanneal** provides SA and SQA with reproducible schedules, sweep-level tracing, and a C++ core with Python bindings.

2 Problem Formulation

2.1 Ising model

Given spins $s_i \in \{-1, +1\}$, the Ising energy is

$$E(\mathbf{s}) = \sum_i h_i s_i + \sum_{i,j} J_{ij} s_i s_j + c. \quad (1)$$

The goal is to minimize $E(\mathbf{s})$.

2.2 QUBO

For binary variables $x_i \in \{0, 1\}$, a quadratic unconstrained binary optimization (QUBO) instance is

$$E(\mathbf{x}) = \sum_{i,j} Q_{ij} x_i x_j. \quad (2)$$

We map QUBO to Ising with $x_i = (1 + s_i)/2$, yielding an equivalent Ising model with transformed h , J , and constant shift c .

3 Simulated Annealing

SA performs a Markov chain over spin configurations. At each temperature T (or inverse temperature β), it proposes single-spin flips and accepts them with Metropolis probability

$$P(\text{accept}) = \min\left(1, e^{-\beta\Delta E}\right). \quad (3)$$

The schedule $\beta_0, \beta_1, \dots$ defines the anneal.

4 Simulated Quantum Annealing

4.1 Transverse-field Ising model

The quantum Hamiltonian is

$$\hat{H} = \sum_i h_i \hat{\sigma}_i^z + \sum_{i,j} J_{ij} \hat{\sigma}_i^z \hat{\sigma}_j^z - \Gamma \sum_i \hat{\sigma}_i^x. \quad (4)$$

Here Γ is the transverse-field strength controlling quantum fluctuations.

4.2 Suzuki–Trotter mapping

SQA uses a path-integral representation by discretizing imaginary time into M slices. This yields a classical Ising model over M replicas with nearest-neighbor coupling along the slice dimension. The effective coupling is

$$J_\perp = \frac{1}{2} \log \left(\coth \left(\frac{\beta\Gamma}{M} \right) \right). \quad (5)$$

The algorithm then performs Monte Carlo updates over the augmented state space.

4.3 Update phases

`qanneal` performs two update phases per schedule step:

- Slice updates: single-spin flips within each slice.
- Worldline updates: collective flips of a spin across all slices.

Both are accepted with Metropolis probability using the effective classical energy.

5 Implementation Architecture

The C++ core defines model classes (`DenseIsing`, `SparseIsing`, `QUBO`), schedulers, and annealers. The `Backend` interface isolates the annealing logic from energy and delta-energy computation. Python bindings are provided via `pybind11`.

5.1 State representation

`State` stores spins as `int8_t` values. `SQAState` stores data in a flattened buffer with layout:

$$\text{index} = (r \cdot M + t) \cdot N + i \quad (6)$$

where r is replica, t is slice, and i is spin.

5.2 Observability

Sweep-level observers record the full state, energy traces, and replica-wise energies. This enables reproducibility, diagnostics, and post-hoc analysis of the anneal.

6 Complexity Considerations

Let N be the number of spins, M the number of slices, R the number of replicas, and S the number of sweeps per step. For dense models, each sweep costs $O(N^2)$; for sparse models, the cost scales with the number of edges. SQA multiplies compute by $M \cdot R$.

7 Experimental Protocol (Template)

To compare algorithms:

- Fix a problem family and size N .
- Use identical schedules and sweep budgets.
- Run multiple seeds and report distributions of energies.
- Use sweep-level traces to study convergence and mixing.

8 Limitations

- CPU-first; CUDA is a planned extension.
- Dense models scale quadratically in memory and time.
- SQA is sensitive to schedule tuning and slice count.

9 Future Work

- CUDA backend for dense and sparse kernels.
- Cluster updates and improved worldline moves.
- MPI-based multi-node replicas.
- Automated schedule tuning and diagnostics.

10 Reproducibility

Each annealer supports explicit RNG seeds. Sweep-level traces can be saved for deterministic replay and analysis. The project ships example scripts and unit tests for baseline verification.