Implementation of the Quantum Galton Board

The Quantum Galton Board (QGB) is a quantum algorithm designed to provide an intuitive and physically grounded demonstration of quantum computing's exponential speedup. It achieves this by creating a quantum circuit that directly replicates the mechanics of a classical Galton board, where balls fall through an array of pegs to form a normal distribution. Instead of simulating individual paths sequentially, the QGB prepares a single quantum state that represents a superposition of all possible trajectories, calculating 2^n paths using resources that scale polynomially.[1]

The 'Quantum Peg': The Foundational Circuit Module

The entire QGB is constructed by replicating a fundamental building block known as the "quantum peg." This module simulates the interaction of a single "ball" with a single peg, creating a 50/50 probability of deflection.[1]

- **Components:** The circuit for one peg requires four qubits: one control qubit (q0) and three working qubits (q1,q2,q3).[1]
- **Implementation Steps:**
  1. **Initialization:** All four qubits are initialized to the $|0\rangle$ state.
  2. **Superposition and 'Ball' Placement:** A Hadamard (H) gate is applied to the control qubit (q0) to create an equal superposition. An X-gate is applied to the central working qubit (q2), which now represents the "ball".[1]
  3. **Controlled-SWAP (Fredkin Gate):** A C-SWAP gate is applied to the working qubits q1 and q2, controlled by q0. This entangles the control qubit with the working qubits, creating a superposition of the ball's potential new positions.[1]
  4. **State Finalization:** A sequence of an inverted CNOT gate and a final SWAP operation completes the module's logic, resulting in a final state where the "ball" is in a superposition of having moved to one of two output channels.[1]

Scaling to a Multi-Level QGB

A full Galton board is constructed by composing these quantum peg modules, with the outputs of one level serving as the inputs for the next.

- **Replication:** The peg module is replicated for each peg in the physical system being modeled.[1]
- **Control Qubit Recycling:** A critical feature for scalability is the management of the control qubit. After each peg interaction, the control qubit must be re-balanced and reset using additional CNOT gates and mid-circuit resets. This allows the single control qubit to be reused for every peg in the circuit, preventing a linear increase in control qubits.[1]
- **Resource Requirements:** An n-level QGB requires:
  - **Qubits:** 2n qubits (2n−1 working qubits and one recycled control qubit).[1]
  - **Gates:** The gate complexity scales polynomially, with an upper bound of 2n2+5n+2 gates.[1]

Generating the Normal Distribution

The QGB's quantum mechanical evolution naturally produces the desired statistical distribution.

- **Superposition of All Paths:** The circuit prepares a single, complex quantum state that is a superposition of all 2n possible trajectories through the board.[1] The amplitudes of the basis states in this final superposition directly correspond to the binomial coefficients that define a normal

distribution.[1]

- **Output and Post-Processing:** Measurement of the output qubits yields a bitstring with exactly one '1' (a one-hot encoding), indicating the final bin for that experimental "shot." To obtain the final distribution, classical post-processing is required to reassign these outputs to integers and scale them to form a recognizable bell curve.[1]

The Biased QGB (B-QGB)

The QGB can be extended from a fixed-purpose circuit to a programmable one capable of generating skewed distributions.

- **Implementation:** The Hadamard gate on the control qubit is replaced with a generic rotation gate, $Rx(\theta)$.[1]
- **Functionality:** By setting the rotation angle θ, one can precisely control the left/right probability at each peg. For example, setting $\theta=2\pi/3$ creates a 75%/25% probability split, generating a skewed distribution instead of a normal one.[1]

The Fine-Grained Biased QGB (Universal Statistical Simulator)

The most advanced implementation allows for maximum programmability, turning the circuit into a "Universal Statistical Simulator".[1]

- **Implementation:** This version allows for a unique bias angle, θi, to be set for *each individual peg*. This is achieved by adding RESET and $Rx(\theta i)$ preparations before each peg module is applied.[1]
- **Complexity:** This fine-grained control increases the gate count to approximately $3n2+3n+1$ due to additional corrective CNOT gates needed to manage the state between pegs.[1]
- **Applications:** This programmability enables the simulation of a wide range of discrete probability distributions, with potential applications in financial modeling, machine learning, and network analysis.[1]

Works cited

1. https://arxiv.org/abs/2202.01735