Efficient Node Lookup in Quantum Decision Diagrams

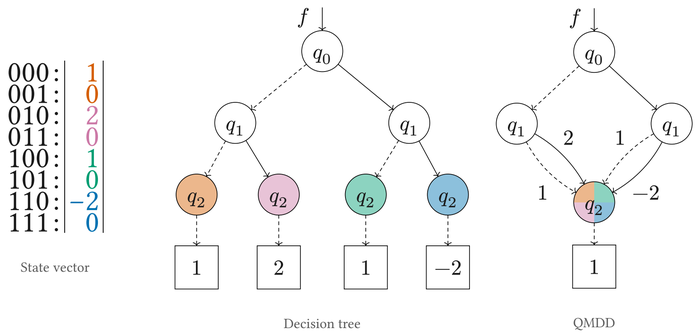
Where lookup cost concentrates and how to exploit it

Author:
Lukas Milieška

Supervisors:
Tim Coopmans & Juul Sanders

1. Problem

Classical simulation validates quantum circuit designs and predicts how algorithms behave on noisy hardware, without a physical quantum computer. But a state on n qubits has 2^n amplitudes. QMDDs compress this by merging nodes that are equal up to a complex factor, stored as edge weights.



- Each new QMDD node is normalised, then **looked up in the unique table** to check whether an equivalent node already exists.
- QolDDer currently stores nodes in a linked list, so every lookup is a linear scan.
- A hash table is a natural replacement, but its design depends on the lookups the circuit actually produces.

Faster unique table lookups directly speed up QMDD simulation.

2. Question

How can node lookups in QMDD simulation be understood, and exploited by unique table design?

1. What collision and probe behaviour do the lookups produce?
2. How does the circuit's structure explain the patterns?
3. How can the unique table be redesigned to exploit them?

The goal is understanding, not only speed.

Key terms

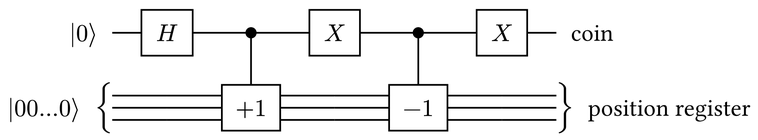
QMDD: Quantum Multiple-Valued Decision Diagram. **Probe**: one slot comparison during a lookup. **Extra probes**: probes beyond the first. **Collision rate**: fraction of operations that need more than one probe. **Same-child node**: both edges share a child. **Direct table**: array indexed using the key, with no probing.

3. Method

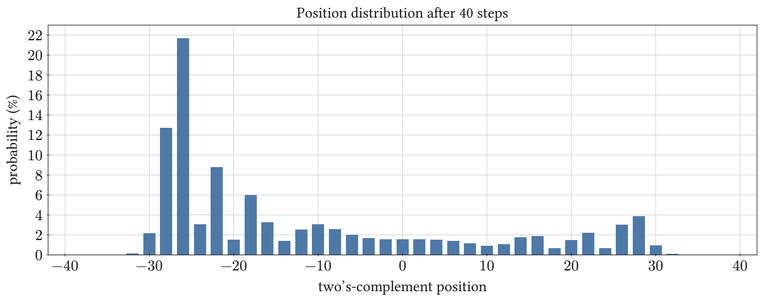
- Instrument every unique table operation: type, qubit, phase, step, and probe count.
- Compare three designs:
  - **Baseline**: hashes the two child nodes only.
  - **Edge-weight**: hashes the full key, including both edge weights.
  - **Latest**: hashes the full key and routes common node classes into direct tables.
- Group probe cost by node class, qubit role, and phase, then explain it using circuit structure.
- White-box check: constant Deutsch–Jozsa has only a few predictable nodes, so a perfect unique table can be designed.

Trace metrics plus circuit structure together explain lookup cost.

4. Benchmark: 1D Quantum Walk



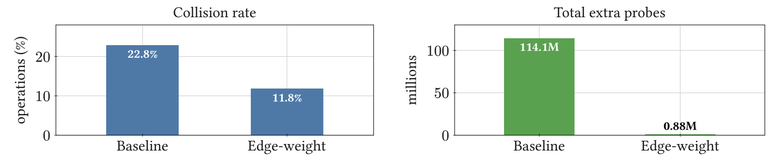
- Per step: coin Hadamard, then $+1$ on the position when coin=1 and -1 when coin=0, then pruning.
- 40 steps, 23 qubits: 1 coin, 12 position, 10 ancilla.
- Nearly 5M unique table operations per run.
- Structured enough for circuit-level interpretation, complex enough for non-trivial QMDDs.



Quantum Walk: a non-trivial circuit, used as the main workload.

5. Result 1: Is collision rate a good measure of cost?

About 2× difference in collision rate, yet total probing cost differs by two orders of magnitude.

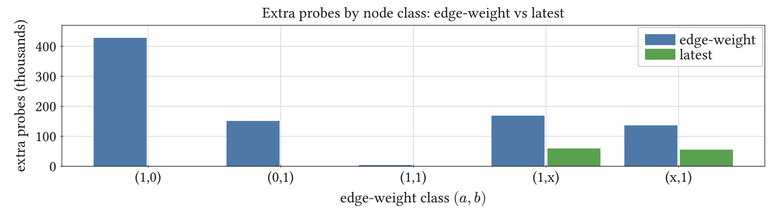


2× change in collision rate but a 130× change in cost

No. A few very long probe runs, not the collision rate, drive the cost.

6. Result 2: Cost lives in a few node classes

Grouping nodes by their normalised edge weights (a, b) shows where probing concentrates.



- Even with full-key hashing, recurring **same-child** nodes $(1, 0)$, $(0, 1)$, $(1, 1)$ still drive most of the cost.
- The **latest** design routes them into direct tables: 79.8% of operations then need no hashing or probing.

0.1871 → 0.0287 extra probes per operation, for 3.17× memory

Tailor the table to recurring node classes.

7. Takeaway and next steps

- Understanding **structure**, not just chasing fewer collisions, nearly removes unique table probing.
- Future work: gate cache, white-box prediction, a second circuit family, and LIMDDs.