Hashing in Quantum Decision Diagrams

Hash-Table Design for QMDD Simulation of Grover's Algorithm with SAT Oracles

Matei-Stefan Aruxandei, EEMCS, TU Delft

Supervisors: Tim Coopmans, Juul Sanders

1 Introduction

Classical simulation of quantum circuits is essential for design and verification, but storing 2^n amplitudes scales infeasibly past 40 qubits. QMDDs share sub-structure in a DAG. Canonicity is enforced by a unique table, a hash table queried at every reconstructed node which is the main source of hash-table traffic during simulation.

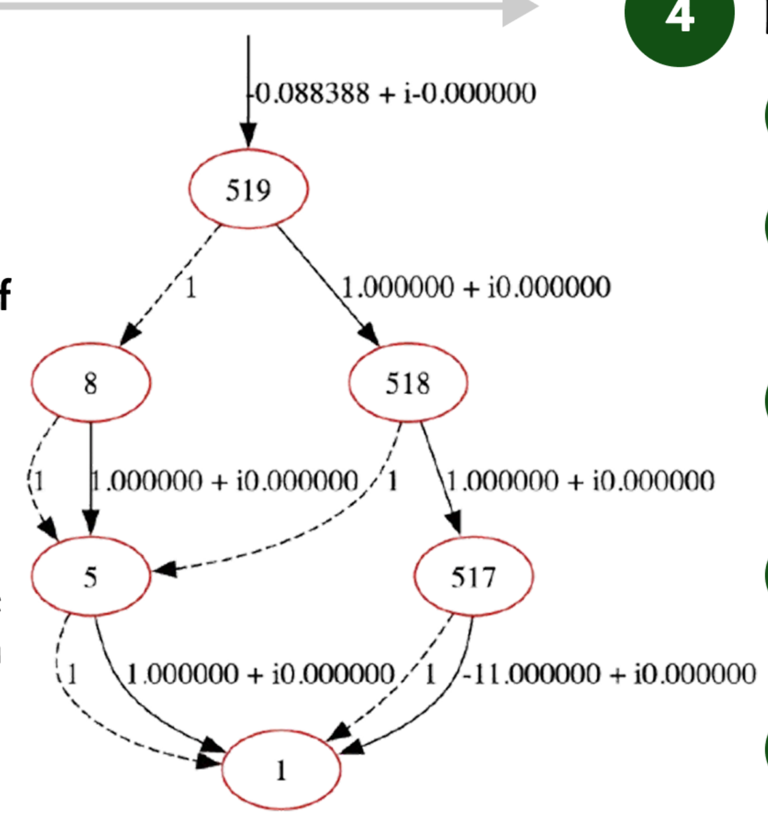
2 Research Question

For Grover's algorithm with SAT oracles, can the access pattern its QMDD simulation induces on the unique table be characterized well enough to design a hash-table scheme that reduces collisions compared to a baseline, at acceptable memory cost?

- (a) How many unique-table collisions does the baseline hashing scheme of the QolDDer-QMDD simulator, chosen without reference to the structure of Grover-SAT circuits, generate as the number of SAT variables grows?
- (b) Which structural properties of Grover-SAT QMDD simulation explain and predict the unique-table key population that causes these collisions?
- (c) Can an alternative hash-table design, motivated by that characterization from (b), reduce collisions relative to the baseline at modest memory cost?

3 Background

A QMDD represents 2^n-amplitude vectors and $2^n \times 2^n$ gate matrices as a DAG with shared sub-structure: each non-terminal vertex is labelled by a qubit, and each edge carries a complex weight. Reading off any entry means walking one path from source to terminal and multiplying the weights along it. Canonicity: no two vertices share the same variable, children, and weights. Low edges, normalised to 0 or 1, guarantee that every distinct substate appears once. The unique table is the hash table that enforces this on each newly reconstructed vertex.



4 Methodology

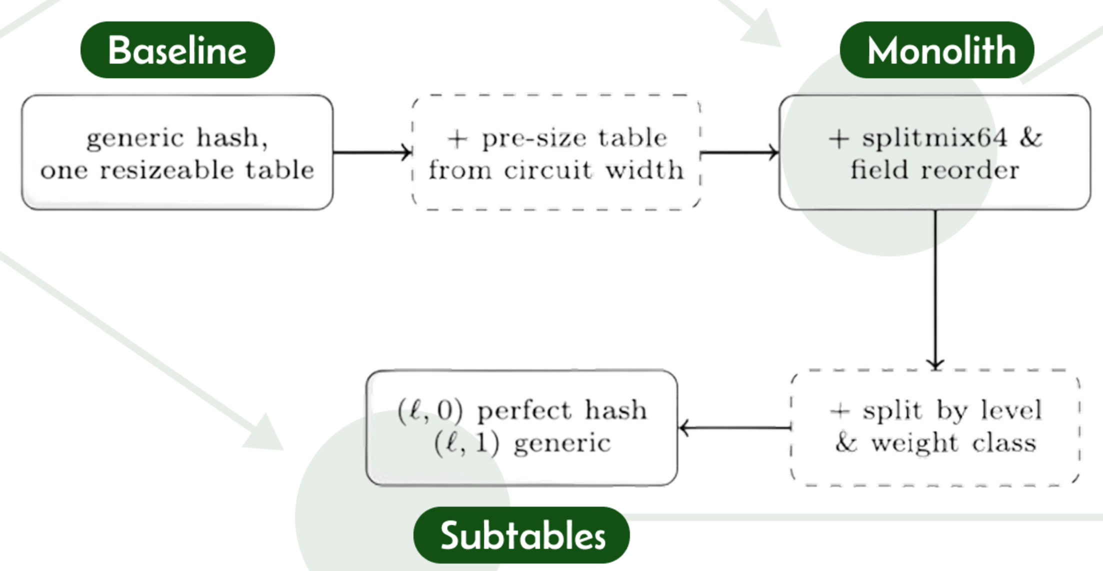
- (a) Simulator: qoldder-qmdd, using a universal gate set.
- (b) Baseline: separate chaining + Fibonacci hashing (multiplier $\lfloor 2^{64}/\phi \rfloor$). Table doubles once load hits 0.75.
- (c) Metrics: collisions, peak memory, lookup distance, per-key hit count. Reference: $E[c] = n - m + m(1 - 1/m)^n$ under uniform hashing.
- (d) Pilot: run Deutsch–Jozsa at n=20 to make sure the tracking infrastructure works and eliminate any hypotheses before moving to Grover-SAT.
- (e) Benchmark: DIMACS → Qiskit PhaseOracle → transpile to {cx, t, tdg, h, x, z} → Grover → simulate for n = 6, 8, 10, 12.

5 Alternative Design

Both designs build on the baseline and keep separate chaining: the garbage collector prunes a large fraction of entries after each Grover iteration, which makes open addressing impractical.

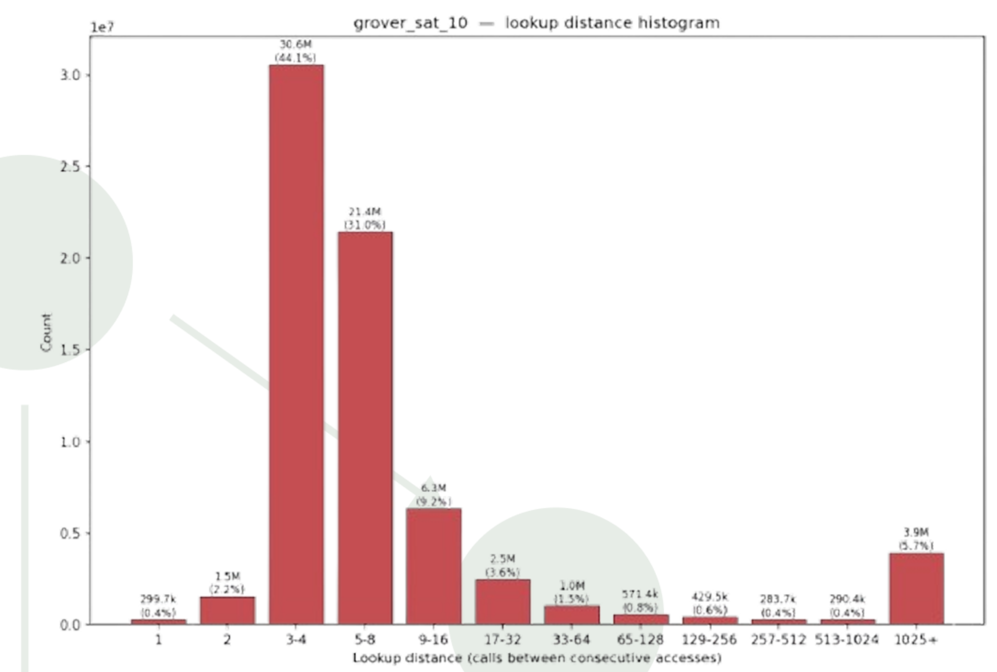
Monolith keeps a single flat table but pre-sizes the bucket array from the qubit count (eliminating most resizes) and replaces Fibonacci hashing with a splitmix64 finalizer for better bit distribution.

Subtables splits keys by QMDD level and weight class, removing cross-level collisions and isolating the predictable, fixed-scalar keys. Because the predictor reconstructs those keys each round, a per-round perfect hash places them with zero collisions, while the drifting keys fall back to the monolith hashing function.

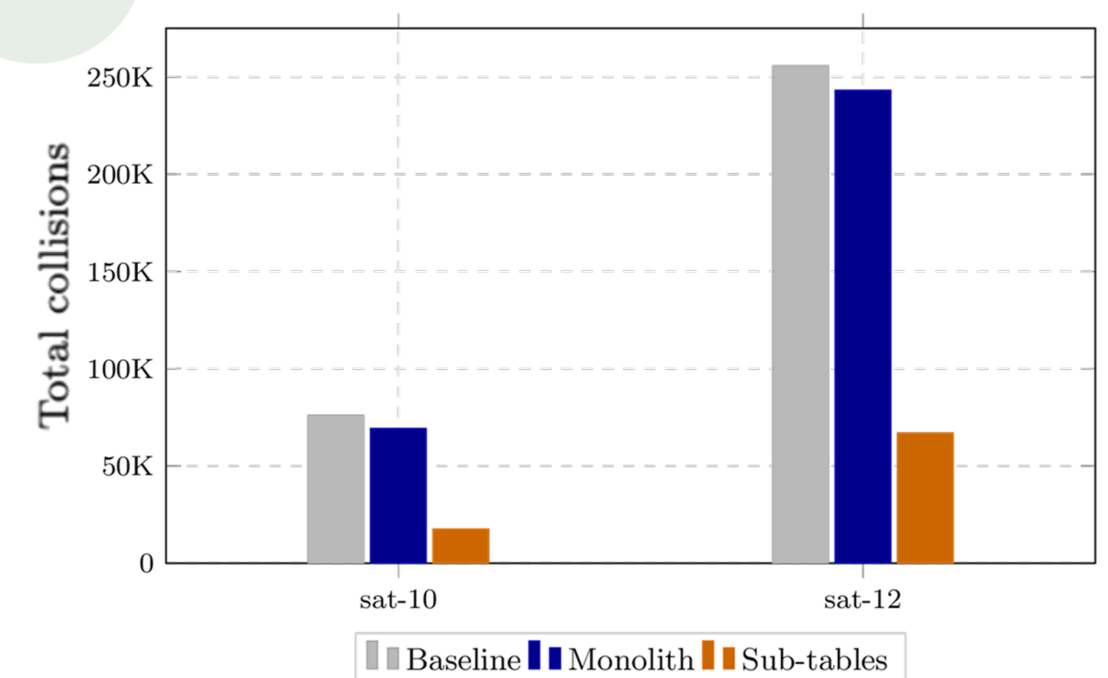


6 Results

A histogram of lookup distances shows strong short-range temporal locality in DJ: 24.6% of lookups re-access the same entry on the very next call, and 65% fall within distance 4, suggesting a most-recent cache would help. On Grover-SAT this locality disappears almost entirely, with distance-1 dropping to 0.4% and mass concentrating at distances 3–8. A cache would catch almost nothing here.



The monolith already lands within 1% of the theoretical per-iteration collision bound, meaning the hash function is essentially optimal. Further gains require structural changes, which is what subtables deliver: because the predictor reconstructs most keys each round, a per-round perfect hash places those predictable keys with zero collisions, cutting collisions by 63–75% on top of the monolith.



The memory overhead is 1.5–23% across all instances, in exchange for 63–75% fewer collisions, which is a favourable trade on every case. The overhead stays small on most instances and reaches 23% only on the largest, where the prediction breaks down near the end.

7 Discussion

Why Subtables Reduce Collisions

- Grover-SAT prunes the table after every iteration (~10:1 churn on sat-12), making the per-iteration bound the right collision model which means churn dominates, not load.
- The monolith matches the uniform-hashing bound within 0.1–1%, so splitmix64 is optimal. Further gains require structural changes.
- Subtables place the predictable keys with a per-round perfect hash and keep the changing keys on a generic hash: 63–75% fewer collisions than the monolith.

Why the Reduction Varies

- sat-8's 64% (vs. ~75% on sat 10) is the lowest because the warmup rounds, before the perfect hash is active, are a larger share of its total number of iterations.

Access Patterns

- V-chain MCZ decomposition concentrates lookups on a narrow ancilla band, producing dominant distances 3–8. The gate cache bounds traversal depth. The 5–8 tail grows as the data region deepens with qubit count.
- On Grover-SAT, DJ locality essentially disappears: distance-1 drops from 24.6% to 0.4%. The mass piles up at distances 3 to 8, which together account for about three quarters of all lookups.

8 Conclusion and Future Work

This work:

- (a) characterizes the Grover-SAT unique-table access pattern, each round rebuilds a near-fixed, predictable per-level key template
- (b) exploits that: placing the predictable keys with a per-round perfect hash cuts collisions 63–75% vs. the monolith, at 1.5–23% memory overhead.

Future work:

- (a) a native multi-controlled-Z oracle removes the ancilla band and spine. The relation classes must be re-derived.
- (b) writing the predicted nodes directly each round, instead of rebuilding them gate by gate and check how the performance changes

(1) D.M. Miller and M.A. Thornton. QMDD: A decision diagram structure for reversible and quantum circuits. In 36th International Symposium on Multiple-Valued Logic (ISMVL'06), page 30, Singapore, 2006. IEEE.
(2) Alwin Zulehner and Robert Wille. Decision diagram-based simulation. In Alwin Zulehner and Robert Wille, editors, Introducing Design Automation for Quantum Computing, pages 37–56. Springer International Publishing, Cham, 2020.
(3) Lov K. Grover. A fast quantum mechanical algorithm for database search. In Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing (STOC '96), pages 212–219, Philadelphia, PA, USA, 1996. ACM.
(4) Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. Introduction to Algorithms. MIT Press, Cambridge, MA, 4 edition, 2022.
(5) Tim Coopmans. QolDDer-QMDD: A QMDD implementation.