Reordering Decision Diagrams for Quantum Computing Is Harder Than You Might Think

Stefan Hillmich[1], Lukas Burgholzer[1], Florian Stögmüller[1], and Robert Wille[2,3]

[1] Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
[2] Chair for Design Automation, Technical University of Munich, Germany
[3] Software Competence Center Hagenberg GmbH (SCCH), Austria
{stefan.hillmich,lukas.burgholzer}@jku.at, robert.wille@tum.de
https://www.cda.cit.tum.de/research/quantum/

Abstract. Decision diagrams have proven to be a useful data structure in both, conventional and quantum computing, to compactly represent exponentially large data in many cases. Several approaches exist to further reduce the size of decision diagrams, i.e., their number of nodes. *Reordering* is one such approach to shrink decision diagrams by changing the order of variables in the representation. In the conventional world, this approach is established and its availability taken for granted. For quantum computing however, first approaches exist, but could not fully exploit a similar potential yet. In this paper, we investigate the differences between reordering decision diagrams in the conventional and the quantum world and, afterwards, unveil challenges that explain why reordering is much harder in the latter. A case study shows that, also for quantum computing, reordering may lead to improvements of several orders of magnitude in the size of the decision diagrams, but also requires substantially more runtime.

1 Introduction

Quantum computers are promising to solve important problems significantly faster than conventional computers ever could. Shor's algorithm [1] and Grover's search [2] are two famous examples, albeit especially Shor's algorithm cannot be handled in a scalable fashion by current quantum computers. Still, there are areas where current quantum hardware can provide an advantage today, such as machine learning [3] and chemistry [4]. This computational power mainly stems from the exploitation of *superposition*, i.e., that quantum states can assume a state in which all basis states are represented at the same time, and *entanglement*, which allows operation on one qubit to also influence other qubits as well. The potential of quantum computers is witnessed by the vast efforts undertaken by companies such as IBM, Google, and Rigetti to build the physical hardware and to develop corresponding design automation tools.

These methods and tools for design automation need to work on conventional hardware but, at the same time, have to deal with the complexity of the quantum world. To this end, representing a quantum state requires an exponential

amount of memory with respect to the number of qubits, often represented as a 2^n-dimensional vector. Rooted in the underlying mathematical principles, the worst-case complexity will always be of this exponential kind [5]. However, utilizing more sophisticated and adaptive data structures such as *decision diagrams* (DDs) [6]–[14] can lead to much more compact representations in many cases.

In the design automation for conventional circuits and systems, decision diagrams have been established since the 90's (see, e.g., [15], [16]). There, it has been shown that the size of decision diagrams significantly depends on the order in which the variables (in case of decision diagrams for quantum computing, the qubits) are encoded. Even though determining an optimal *variable order* is a coNP-hard problem [17], the potential reductions in size motivated a plethora of reordering schemes aimed at determining suitable or even the best possible variable orders for a given decision diagram [17]–[23].

The success of reordering in the conventional design automation raises the question, whether similar approaches are suitable for the quantum world as well. A few attempts of employing reordering schemes for decision diagrams for quantum computing have been made [11], [24]–[26]. However, those attempts have remained rather rare and considered only a prototypical level with small examples yet. Moreover, implementations of decision diagrams such as provided in [11] frequently abort when reordering is applied to larger examples due to inaccuracies in the floating-point numbers storing the edge weights. This necessitates an investigation on the challenges that have prevented a fully-fledged application of reordering in decision diagrams for quantum computing thus far.

In this work, we investigate the challenges of reordering which emerge during the implementation of this feature for decision diagrams with complex edge weights. As the challenges arise due to inaccuracies in the floating-point representation of real numbers, all types with complex edge weights, such as QMDDs [7] and LIMDDs [13], are susceptible. To this end, it becomes apparent that implementing reordering decision diagrams for quantum computing is much harder than originally thought considering the simplicity of the concept itself. Using this knowledge, we present a solution that handles those challenges and evaluate how this eventually affects the performance of reordering. Our case study shows that reordering may allow for substantial improvements (in some cases yielding decision diagrams which are several orders of magnitudes smaller in their size), but also will require substantially more runtime—showing that designers should decide whether reordering pays off in their use case. For the first time, this explains the reluctance of using reordering in decision diagrams for quantum computing, but also shows the potential still available in this optimization scheme.

The remainder of this paper is organized as follows: Section 2 briefly reviews the basics of quantum computing and decision diagrams. Section 3 explains the concepts of reordering in decision diagrams, whereas Section 4 discusses challenges that arise when implementing reordering in decision diagrams for quantum computing together with a corresponding solution. In Section 5, we present the results of our case study. Finally, we conclude the paper in Section 6.

2 Background

In order to keep the work self-contained, we briefly review the basics on quantum computing and decision diagrams in this section.

2.1 Quantum Computing

In quantum computing, the basic unit of information is the *quantum bit* or *qubit* [5], [27]. As a conventional bit, it can assume the corresponding computational basis states $|0\rangle$ and $|1\rangle$ (in Dirac notation). However, qubits can additionally assume linear combinations of the basis states. They are, justifiably, in both states at the same time. A more precise notion is $|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle$ where $\alpha_0, \alpha_1 \in \mathbb{C}$ are referred to as *amplitudes*. If both α_0 and α_1 are non-zero, the quantum state is said to be in *superposition*. Additionally, in systems with more than one qubit, the quantum state can be *entangled*, meaning that an operation on one qubit may affect other qubits as well.

The amplitudes are fundamentally opaque in a physical quantum computer. The only way to retrieve information on a quantum state is *measurement*. Measurement is probabilistic and results in a single basis state while, at the same time, superposition and entanglement are destroyed. The probability to measure any basis state is determined by its amplitude: Given α_i, the squared magnitude $|\alpha_i|^2$ is the probability to measure the basis state $|i\rangle$. Therefore, the amplitudes of the quantum state are constrained such that the sum of the squared magnitudes must equal one—referred to as *normalization constraint*. For a one-qubit system $|\alpha_0|^2 + |\alpha_1|^2 = 1$ must hold. Quantum states with n qubits have 2^n basis states, each with a corresponding amplitude. Multi-qubit states are also subject to the normalization constraint $\sum_{i \in \{0,1\}^n} |\alpha_i|^2 = 1$. Commonly, quantum states are represented as 2^n-dimensional vectors containing the amplitudes, implemented as arrays of floating-point numbers.

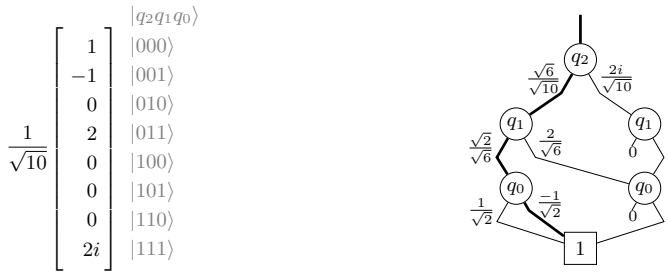
Example 1. Consider the pure two-qubit quantum state $|\psi\rangle$, which is set to

$$|\psi\rangle = 1/\sqrt{2} \cdot |00\rangle + 0 \cdot |01\rangle + 0 \cdot |10\rangle + 1/\sqrt{2} \cdot |11\rangle\,.$$

This state is valid, since $|1/\sqrt{2}|^2 + |0|^2 + |0|^2 + |1/\sqrt{2}|^2 = 1$ satisfies the normalization constraint. As a vector, the state is written as $|\psi\rangle = \left[\, 1/\sqrt{2} \;\; 0 \;\; 0 \;\; 1/\sqrt{2} \,\right]^{\mathrm{T}}$. Due to the superposition, measuring this state yields either of the two basis states $|00\rangle$ or $|11\rangle$ with a probability of $|1/\sqrt{2}|^2 = 1/2$ each. After the measurement, the superposition is destroyed and the quantum state is fixed to the measured state, i.e., subsequent measurements yield the same result.

2.2 Decision Diagrams

Quantum states require 2^n-dimensional vectors to represent n qubits if this straightforward representation is chosen. In many cases *decision diagrams* (DDs) can drastically reduce this exponential complexity by exploiting redundancies [6], [8]–[12], although the worst-case complexity remains exponential.



(a) Vector (b) DD

Fig. 1. Two quantum states with vector and DD representation, respectively

The structural redundancies in vectors can be exploited by shared structures in decision diagrams. More precisely, the vector is split into equally sized upper and lower sub-vectors. There are n levels of splitting for an n-qubit state until the individual elements are reached. If identical sub-vectors occur in this procedure, they are detected and represented by shared nodes. The consistent application of a *normalization scheme* guarantees a canonical representation of quantum states and thus maximally compact decision diagrams (given a fixed variable ordering). In the resulting decision diagram, the amplitudes are encoded in the edge weights. To get the amplitude of a basis state the edge weights along the corresponding path have to be multiplied.

Example 2. Consider the state vector in Fig. 1a. The annotations on the right denote the basis state each amplitude corresponds to. In Fig. 1b, a decision diagram representing the same state is depicted with the normalization introduced in [28]. To access the amplitude of basis state $|001\rangle$, the bolded path in the decision diagram has to be traversed and the edge weights along this path have to be multiplied, e.g., ($q_2 = 0$, $q_1 = 0$, $q_0 = 1$) yielding $1 \cdot \sqrt{6}/\sqrt{10} \cdot \sqrt{2}/\sqrt{6} \cdot -1/\sqrt{2} = -1/\sqrt{10}$.

3 Reordering Decision Diagrams

This section reviews the effect of the variable order in decision diagrams and the conceptual approach to reordering. To this end, we take the findings from reordering decision diagrams in the conventional world (e.g., from [17], [19]–[23]) and adapt them to the corresponding representation of quantum states. Afterwards, we use this as basis to show that certain corner cases frequently occur in reordering of decision diagrams for quantum computing—providing an explanation why reordering has been investigated only in a theoretic or prototypical fashion in the quantum world (e.g., in [10], [11], [24]–[26]).

The compaction decision diagrams can achieve significantly depends on the order in which the variables (or qubits) are represented. In fact, the *variable order* has a great influence on the size of the decision diagram [17] and, in particular cases, can be the difference between a compact and an exponential representation

(a) DD with order $q_2 < q_1 < q_0$ (b) DD with order $q_1 < q_2 < q_0$

Fig. 2. Variable swap between q_2 and q_1 decreases the number of nodes

with respect to the number of variables/qubits. The variable order is denoted as $q_{i_0} < q_{i_1} < \ldots < q_{i_{n-1}}$ for a system with n variables/qubits and orders the variables/qubits in the decision diagram from the root node at the top to the qubit that appears in the last level before the terminal node.

Example 3. Consider the decision diagrams in Fig. 2, both of which represent the same quantum state. In (a), the application of variable order $q_2 < q_1 < q_0$ yields a decision diagram with five non-terminal nodes whereas, in (b), the variable order $q_1 < q_2 < q_0$ yields a decision diagram with only four non-terminal nodes—a 20 % reduction of non-terminal nodes.

Changing the variable order is termed *reordering* [19]–[23]. The simplest change in the variable order is exchanging variables/qubits that are adjacent in the current variable order. For two adjacent variables/qubits $q_{i+1} < q_i$, this requires swapping the "inner edges" representing $|q_{i+1}q_i\rangle = |01\rangle$ and $|q_{i+1}q_i\rangle = |10\rangle$. Keeping in mind that the decision diagrams for quantum states represent vectors, this swap corresponds to the following transformation:

$$\begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \begin{matrix} |q_{i+1}q_i\rangle \\ |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix} \xrightarrow[q_{i+1} \text{ and } q_i]{\text{swap}} \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} \begin{matrix} |q_iq_{i+1}\rangle \\ |00\rangle \\ |\mathbf{10}\rangle \\ |\mathbf{01}\rangle \\ |11\rangle \end{matrix} \xrightarrow[\text{indices}]{\text{sort}} \begin{bmatrix} A \\ C \\ B \\ D \end{bmatrix} \begin{matrix} |q_iq_{i+1}\rangle \\ |00\rangle \\ |01\rangle \\ |10\rangle \\ |11\rangle \end{matrix} .$$

In this description, A, B, C, and D can be complex numbers (if q_{i+1} and q_i are the only qubits) or sub-vectors themselves (if the system has more qubits). Of course, there may be multiple nodes labeled q_i. In this case, these the variable swapping has to be applied to each such node.

Example 4. The general simplicity of swapping two adjacent variables/qubits in a decision diagram is illustrated in Fig. 3. Only the inner outgoing edges on the lower level need to be swapped and the node labels have to be exchanged.

Changing the variable order from one to another is realized by iteratively exchanging adjacent variables/qubits in the current variable order until the desired

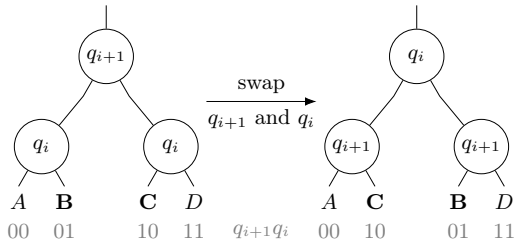


Fig. 3. Conceptually swapping two variables/qubits

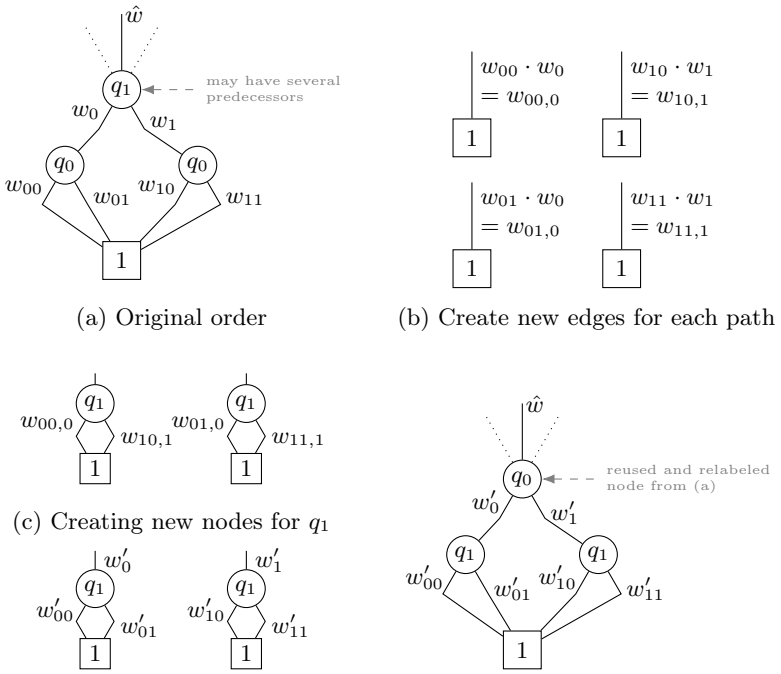
variable order is attained. Doing so is also commonly required by heuristics trying to a find good variable order. Finding the minimal solution is an coNP-hard problem [17] and often done via exhaustive search.

In the conventional world, reordering is a standard approach to reduce the size of decision diagrams such as BDDs [17], [19]. A well-known reordering heuristic is *sifting*, which has a quadratic complexity in the number of variables/qubits [23]. This approach repeatedly selects a qubit and moves it up and down in the decision diagram to find the minimal position for said variable/qubit. Hence, for a decision diagram with n variables/qubits, there are $n - 1$ positions to consider for each variables/qubit—yielding an efficient heuristic which often yields good enough results. Besides that, a plethora of further works exist which aim to determine good orders or even try to obtain the best possible order as efficiently as possible (see, e.g., [18], [20]–[23]).

4 Challenges in Reordering Decision Diagrams for Quantum Computing

In the quantum world, decision diagrams recently were established as a data structure to efficiently handle quantum states and quantum function descriptions for simulation, synthesis, and verification. However, while reordering is a tried and tested procedure for conventional decision diagrams, it has been rarely used in decision diagrams for quantum computing yet. In fact, reordering for the quantum world has been considered only on a rather conceptual level with small examples [11], [25], [26]. In this section, we discuss why this might be the case and particularly show the challenges of reordering which emerge in quantum computing when more complex decision diagrams are considered.

Conceptually, reordering decision diagrams of quantum states is conducted similarly to reordering in conventional decision diagrams and consists of one or more variable swaps as illustrated in Fig. 3: Each variable swap involves two adjacent levels swapping their inner out-going edges on the lower level and accordingly relabeling the nodes. However, while this is a conceptually simple procedure and easy to realize for conventional decision diagrams, a corresponding realization for the quantum world has to consider the increased complexity. Indeed, decision diagrams for quantum computing additionally need to represent complex



(a) Original order

(b) Create new edges for each path

(c) Creating new nodes for q_1

(d) Normalizing nodes

(e) Swapped order after relabeling

Fig. 4. Step-by-step variable swap for a single node q_1

numbers, which are commonly encoded in the edge weights (see Section 2.2). This may lead to severe challenges that have not been considered thus far and are investigated in the following.

4.1 Floating-Point Accuracy

Quantum computing uses complex numbers to describe states and functions, so decision diagrams for quantum have to incorporate complex numbers as well ($1/\sqrt{2}$ is an example of a common irrational factor in quantum computing occurring in the real or imaginary parts of a complex number). For design automation tasks on conventional computers, we are left with two choices: Do calculations symbolically to retain absolute accuracy or use some form of approximation, such as floating-point numbers. Exact representations, such as the algebraic representation proposed in [29], are too computationally expensive for all but small benchmarks. In the case of [29], which uses tuples of integers for representation, the integers quickly become larger than natively representable with 64 bits and the implementation therefore introduces an overhead by using arbitrary large integers. Hence, most implementations use floating-point numbers. However, the edge weights encoding the amplitudes of the quantum state are affected by most operations, such as multiplying edge weights of decision diagrams in simulation and reordering. Thus, due to the limited accuracy of floating-point numbers, each modification of the edge weights carries the risk of losing tiny bits of information.

Example 5. Fig. 4 illustrates the required computations on the edge weights, when swapping two adjacent qubits in a decision diagram. As illustrated in Fig. 4a (showing the original order) and in Fig. 4e (showing the resulting order), the procedure follows the original idea from the conventional world (see Fig. 3) and just adjusts the inner outgoing edges and the node labels. In addition to that, however, the edge weights also need to be adjusted when decision diagrams for quantum computing are considered. This is particularity shown in Fig. 4b and 4d—and provides a challenge to be addressed. In fact, multiplying numbers that are not exactly representable with floating-point numbers, such as $1/\sqrt{2}$ or $1/3$, will result in a product that might be slightly off its real value. This difference in actual number and exact number interferes with the detection of identical sub-structures and, hence, leads to larger decision diagrams.

Most state-of-the-art implementations tackle the challenge of lost sharing due to floating-point inaccuracies by employing some form of tolerance when comparing floating-point numbers Commonly, two complex numbers a and b are regarded as equal if $|\operatorname{Re}(a) - \operatorname{Re}(b)| < T$ and $|\operatorname{Im}(a) - \operatorname{Im}(b)| < T$ for some tolerance T. However, using such a tolerance to mitigate floating-point inaccuracies causes another challenge when trying to do reordering.

4.2 Node Collisions

The inaccuracy of floating-point numbers and the common usage of tolerances for comparisons eventually creates a more severe problem for reordering nodes.

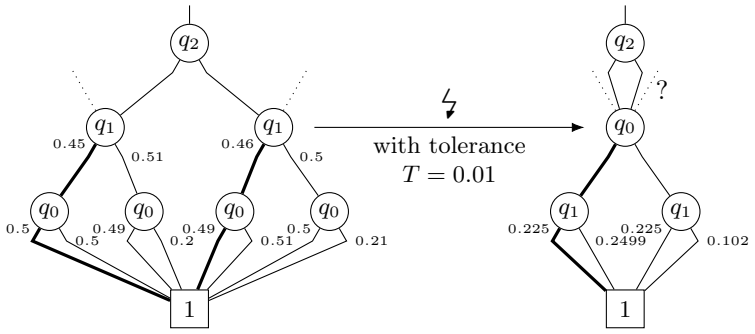


Fig. 5. Decision diagram where exchanging q_1 and q_0 leads to a collision

In theory, doing a variable swap between q_{i+1} and q_i does not change the number of nodes on the level of q_{i+1}. However, in practice, a loss of precision may cause two hitherto different nodes to become identical.

Example 6. Consider the decision diagram shown in Fig. 5 and assume that a tolerance of $T = 0.01$ is applied. Now, the qubits q_1 and q_0 should be swapped. Note that both nodes labeled q_1 are very similar before the variable swap, but the edge weights differ by at least the tolerance. During the swapping procedure, weights of outgoing edges from the left-hand side of q_1's successors are multiplied as shown in Fig. 4b. Because the two q_1-nodes were similar to begin with, the products from the left-hand side q_1-node will be similar to the products from the right-hand side q_1-node as well. During the next steps (see Fig. 4c) this may lead to a situation where, due to the applied tolerance, nodes will be considered identical during lookup of existing nodes to exploit redundancies—a *collision* occurs. More precisely, consider the left successors in the decision diagram shown in Fig. 5 as an example (bolded in the figure). Here, we get the products $0.45 \cdot 0.5 = 0.225$ and $0.46 \cdot 0.49 = 0.2254$. Due to the tolerance, both weights will be treated as identical since $|0.225 - 0.2254| < 0.01$—the same holds for the remaining products. As a result, the resulting nodes will be considered equal during lookup of existing nodes.

At a first glance, *collisions* (a term borrowed from unwanted collisions in hash tables) may seem desirable since the number of nodes is reduced. In fact, the number of nodes in a fully populated reduced decision diagram is invariant to the variable ordering (assuming sufficient accuracy in represented numbers), whereas collisions possibly reduce the number of nodes. However, since the nodes on the upper level of the variable swap are re-used to avoid reconstructing the whole decision diagram, this invalidates all edges pointing to the colliding node. Worse yet, keeping nodes regardless of a collision creates a conflict with the commonly used *unique table*. The unique table holds pointers to all inserted nodes and is essential to efficiently identifying identical sub-structures for sharing. As the name suggests, it is supposed to hold unique entries—forcibly inserting duplicates creates unreachable nodes.

Example 6 (Continued). Consider again Fig. 5. Edges pointing to the left-hand side q_1-node in the original decision diagram would work as expected in the reordered version. However, edges pointing to the right-hand side q_1-node in the original decision diagram are invalid in the reordered version. The resulting decision diagram would not correctly represent the state anymore if the collision were ignored.

Previous works (such as [11], [24]–[26]) did not consider collisions and, hence, failed when node collisions occurred. This may be due to collisions only occurring with increasingly complex decision diagrams and previous works considered rather small examples in their evaluations. In cases where an implementation is available (e.g., [11]), the execution is aborted with an error message if a node collision happens. However, our case study in Section 5 confirms that collisions are a common problem in more complex quantum states and, hence, need to be considered.

We propose to mitigate the problem of colliding nodes by readjusting edges pointing to the collided node. To this end, the corresponding parts of the unique table are scanned to find the address of the collided node and substitute the address of the original node. Further, the decision diagram may contain nodes that are yet to be inserted into the unique table and, thus, the current decision diagram is scanned as well. Note that neither scan approach is sufficient: Multiple decision diagrams can exist in the same unique table to increase the sharing potential, so nodes outside the current decision diagram may be affected.

Depending on the structure above the collision, it may be necessary to perform cascading substitutions up to one level below the root node. Handling node collisions, especially when they cascade, drastically increase the overhead of reordering since parts of the decision diagram have to be reconstructed. In addition, the merging of nodes to handle collisions is an irreversible operation. The evaluations in the next section show how this eventually makes the entire reordering process harder with respect to runtime. Before, however, another challenge is considered.

4.3 Normalization

Finally, the canonicity of decision diagrams with the same variable order should be preserved. While canonicity might not be a must-have requirement for an efficient data structure (e.g., in tasks such as simulation or synthesis), it is essential in design tasks such as equivalence checking. However, even if it is not required in tasks such as simulation, canonicity increases the likelihood of detecting identical sub-structures—resulting in a more compact decision diagram.

In conventional BDDs, canonicity is achieved by fixing the variable order and ensuring there are no two different nodes representing the same sub-structure—the decision diagrams have to be *ordered* and *reduced* [17]. For decision diagrams in the quantum world, edge weights have to be considered additionally. Here, edge weights are *normalized* to guarantee canonicity. Unfortunately, normalization schemes may cascade upwards through the decision diagram and thus limit

efficiency of this desirable, if not required, operation during reordering. The introduction of *normalization factors* in the nodes as explained in [11] mitigates the premature normalization through the whole decision diagram during the reordering procedure. Nonetheless, after the reordering is performed, the normalization factors have to be applied to the out-going edges with subsequent normalization through the whole decision diagram. This may significantly increase the runtime required to conduct reordering.

5 Case Study

The investigations from above showed that, although core principles can be re-used from the conventional world, reordering of decision diagrams is significantly harder in the quantum world. This raises the question of whether this optimization technique, which is well known and established for conventional decision diagrams, is still applicable on a larger scale for decision diagrams for quantum computing as well. To evaluate the applicability, we conducted a case study in which we considered quantum benchmarks such as the Quantum Fourier Transform [30], Grover's search [2], and Google's quantum-supremacy benchmarks [31] (using conditional phase-gates) and studied the effect of reordering corresponding decision diagrams representing the resulting quantum states from them with a tolerance $T = 10^{-13}$. Additionally, the reordering was only conducted on decision diagrams with at least 1000 nodes and less than 90 % of a complete decision diagram (i.e., $0.9 \cdot 2^{n-1}$ nodes with n denoting the number of qubits). The evaluations were performed on a server running GNU/Linux using an AMD Ryzen 9 3950X and 128 GiB main memory with GNU parallel [32] to orchestrate the execution. The implementation is based on [11], [12], [33] and extended by the schemes proposed in Section 4 to address the investigated challenges and available at `https://github.com/cda-tum/ddsim/tree/reordering` under the MIT license.

The results are listed in Table 1 and discussed in the following. The table's first columns list the benchmarks as well as the number of qubits using the following notation:

- "qft_A" denotes the Quantum Fourier Transform with A qubits,
- "grover_A" denotes Grover's algorithm with A being the size of the oracle, and
- "inst_AxB_C_D" denotes a quantum-supremacy circuit on an $A \times B$ grid with C cycles and D being a running number from `https://github.com/sboixo/GRCS/` to unambiguously identify individual benchmarks.

The following column lists the runtime needed to simulate the corresponding benchmark and, by this, obtain the desired state as a decision diagram. Afterwards, we applied a reordering scheme (namely sifting as reviewed in the end of Section 3) to optimize the resulting decision diagrams. The respectively needed runtime for that as well as the minimal number and maximal number of obtained

Table 1. Effect of Reordering for Quantum States

Benchmark	Qubits	Simulation	Simulation + Reordering (Sifting) of Resulting Quantum State					
		Time [s]	Time [s]	Min. Nodes	Max. Nodes	Abs. Diff.	Rel. Size	Collisions
qft_21	21	1.2	1.7	22	22	0	1.000	0
qft_23	23	1.5	1.8	24	24	0	1.000	0
qft_24	24	1.5	2.6	25	25	0	1.000	0
qft_25	25	1.1	1.5	26	26	0	1.000	0
grover_11	12	0.4	0.5	23	812	789	0.028	153
grover_14	15	0.5	0.8	29	2 700	2 671	0.011	508
grover_17	18	3.1	41.5	35	41 659	41 624	0.001	7 559
grover_20	21	518.5	1 808.2	41	156 414	156 373	<0.001	34 527
grover_22	23	4 584.3	*Timeout*	–	–	–	–	–
grover_23	24	11 313.4	*Timeout*	–	–	–	–	–
inst_4x4_10_0	16	3.0	164.0	54 174	64 445	10 271	0.841	3 440
inst_4x4_10_1	16	1.7	60.7	24 849	61 846	36 997	0.402	3 364
inst_4x4_11_0	16	6.7	148.6	49 925	65 462	15 537	0.763	2 269
inst_4x4_11_1	16	2.0	23.9	24 617	36 909	12 292	0.667	120
inst_4x4_12_0	16	9.7	242.9	65 536	65 536	0	1.000	0
inst_4x4_12_1	16	1.3	48.9	32 769	49 152	16 383	0.667	0
inst_4x4_13_0	16	12.2	210.6	65 536	65 536	0	1.000	0
inst_4x4_13_1	16	1.9	177.2	65 536	65 536	0	1.000	0
inst_4x4_14_0	16	15.8	262.7	65 536	65 536	0	1.000	0
inst_4x4_14_1	16	3.2	158.3	36 697	65 536	28 839	0.560	5 480
inst_4x4_15_0	16	23.1	282.0	65 536	65 536	0	1.000	0
inst_4x4_16_0	16	29.3	282.3	65 536	65 536	0	1.000	0
inst_4x4_16_1	16	9.0	93.3	32 871	65 536	32 665	0.502	2 007
inst_4x5_10_0	20	1 302.5	*Timeout*	–	–	–	–	–
inst_4x5_10_1	20	1 007.1	*Timeout*	–	–	–	–	–

Timeout of simulation and sifting combined was set to 24 h.

nodes are listed in Columns 4–6 of Table 1. The remaining columns provide the corresponding absolute and relative difference in the number of nodes as well as the number of collisions that occurred during this process (see Section 4.2).

Firstly, the results clearly show that the effect of reordering really depends on the considered benchmark. For example, the number of nodes needed to represent the QFT-state is always linear with respect to the number of qubits for all considered orders (including the minimal and maximal cases). This is in-line with observations from the conventional world (where, e.g., functions like AND, OR, etc. are also oblivious to the variable order). On the other hand, there are benchmarks where the applied order is essential for a compact representation; most notably shown by the benchmark "grover_20" which, according to the applied order, either may require close to 160 000 nodes (maximal case) or can be represented by just 41 nodes (minimal case)—a difference of several orders of magnitude. Also for the quantum-supremacy benchmarks substantial optimizations can be achieved in some cases, despite the fact that these benchmarks are designed to contain little to no redundancy and, therefore, are considered worst-case scenarios for decision diagrams. These benchmarks also showcase a consequence of collisions: A fully populated decision diagram in theory will remain fully populated regardless of the reordering. In practice however, given the limited accuracy of floating point numbers, collisions may decrease the number of nodes as can be seen for the benchmark "inst_4x4_16_1".

Secondly, the results confirm that reordering of decision diagrams is very much a time-consuming task in the quantum world. Just applying the heuristic sifting scheme on the considered state representations already required substantial computation times (see Column 4 in Table 1) which frequently exceed the runtime needed to generate the state by simulation in the first place (see Column 3 in Table 1). That is, in contrast to conventional decision diagrams, designers really should consider the trade-off between the runtime of reordering and the size of the decision diagram. In most cases of quantum circuit simulation based on decision diagrams fixing the variable order in the beginning is the preferable approach. A rough guideline favors positioning the control qubits on a lower index compared to the position of the target qubits and minimizing the distance between control index and target index.

Finally, the results provide evidence that, indeed, reordering in decision diagrams for quantum computing is harder than originally thought: Challenges such as the node collisions discussed in Section 4.2 (whose handling causes a significant portion of the increased computation time) are not rare corner cases, but frequently occur (see Column 9 in Table 1). While previous work such as [11], [24]–[26] did not consider collisions (leading to decision diagrams where reordering only works for small examples and/or whose execution is aborted with an error message), the solution presented and evaluated in this work shed light on this.

6 Conclusions

The size of decision diagrams significantly depends on the order in which the corresponding variables/qubits are encoded. Changing the variable order, i.e., reordering, is a tried and tested technique to compact decision diagrams in the conventional world. In the quantum world, however, a similar potential has not been exploited yet. In this paper, we investigated why this might be the case and unveiled the challenges that arise in reordering for quantum decision diagrams. Our findings show that reordering in the quantum world indeed is harder compared to conventional decision diagrams—explaining why previous implementations could not handle reordering of larger decision diagrams. A case study eventually confirms that reordering may lead to improvements of several orders of magnitude although it requires substantially more runtime.

Acknowledgments

This work received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No. 101001318), was part of the Munich Quantum Valley, which is supported by the Bavarian state government with funds from the Hightech Agenda Bayern Plus, and has been supported by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG).

References

[1] P. W. Shor, "Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer," *SIAM Jour. of Comp.*, vol. 26, no. 5, pp. 1484–1509, 1997.

[2] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Symp. on Theory of Computing*, 1996, pp. 212–219.

[3] D. Riste, M. P. da Silva, C. A. Ryan, A. W. Cross, A. D. Córcoles, J. A. Smolin, J. M. Gambetta, J. M. Chow, and B. R. Johnson, "Demonstration of quantum advantage in machine learning," *npj Quantum Information*, vol. 3, no. 1, pp. 1–5, 2017.

[4] Y. Cao, J. Romero, J. P. Olson, *et al.*, "Quantum chemistry in the age of quantum computing," *Chemical reviews*, vol. 119, no. 19, pp. 10 856–10 915, 2019.

[5] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information (10th Anniversary edition)*. Cambridge Univ. Press, 2016.

[6] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Algebraic decision diagrams and their applications," in *Int'l Conf. on CAD*, 1993, pp. 188–191.

[7] D. M. Miller and M. A. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits," in *Int'l Symp. on Multi-Valued Logic*, IEEE Computer Society, 2006, p. 30.

[8] A. Abdollahi and M. Pedram, "Analysis and synthesis of quantum circuits by using quantum decision diagrams," in *Design, Automation and Test in Europe*, 2006, pp. 317–322.

[9] S.-A. Wang, C.-Y. Lu, I.-M. Tsai, and S.-Y. Kuo, "An XQDD-based verification method for quantum circuits," *IEICE Trans. Fundamentals*, vol. 91-A, no. 2, pp. 584–594, 2008.

[10] G. F. Viamontes, I. L. Markov, and J. P. Hayes, *Quantum Circuit Simulation*. Springer, 2009.

[11] P. Niemann, R. Wille, D. M. Miller, M. A. Thornton, and R. Drechsler, "QMDDs: Efficient quantum function representation and manipulation," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 35, no. 1, pp. 86–99, 2016.

[12] A. Zulehner and R. Wille, "Advanced simulation of quantum computations," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 38, no. 5, pp. 848–859, 2019.

[13] L. Vinkhuijzen, T. Coopmans, D. Elkouss, V. Dunjko, and A. Laarman, "LIMDD A decision diagram for simulation of quantum computing including stabilizer states," *CoRR*, vol. abs/2108.00931, 2021.

[14] X. Hong, X. Zhou, S. Li, Y. Feng, and M. Ying, *A tensor network based decision diagram for representation of quantum circuits*, 2021. arXiv: 2009.02618 [quant-ph].

[15] R. E. Bryant, "Symbolic Boolean manipulation with ordered binary-decision diagrams," *ACM Computing Surveys*, vol. 24, no. 3, 1992.

[16] R. I. Bahar, E. A. Frohm, C. M. Gaona, G. D. Hachtel, E. Macii, A. Pardo, and F. Somenzi, "Algebric decision diagrams and their applications," *Formal Methods in System Design*, vol. 10, no. 2-3, pp. 171–206, 1997.

[17] R. E. Bryant, "Graph-based algorithms for boolean function manipulation," *IEEE Transactions on Computers*, vol. C-35, no. 8, pp. 677–691, 1986.

[18] S. J. Friedman and K. J. Supowit, "Finding the optimal variable ordering for binary decision diagrams," in *Design Automation Conf.*, 1987.

[19] R. Rudell, "Dynamic variable ordering for ordered binary decision diagrams," in *Int'l Conf. on CAD*, 1993, pp. 42–47.

[20] N. Ishiura, H. Sawada, and S. Yajima, "Minimization of binary decision diagrams based on exchanges of variables," in *Int'l Conf. on CAD*, 1991, pp. 472–473.

[21] C. Meinel and A. Slobodova, "Speeding up variable reordering of OBDDs," in *Int'l Conf. on Computer Design VLSI in Computers and Processors*, 1997, pp. 338–343.

[22] F. Somenzi, "Efficient manipulation of decision diagrams," *Int'l Journal on Software Tools for Technology Transfer*, vol. 3, no. 2, 2001.

[23] C. Meinel, F. Somenzi, and T. Theobald, "Linear sifting of decision diagrams and its application in synthesis," *IEEE Trans. on CAD of Integrated Circuits and Systems*, vol. 19, no. 5, pp. 521–533, 2000.

[24] S. V. Schaick and K. B. Kent, "Analysis of variable reordering on the QMDD representation of quantum circuits," in *Euromicro Conf. on Digital System Design*, 2007, pp. 347–352.

[25] D. M. Miller, D. Y. Feinstein, and M. A. Thornton, "QMDD minimization using sifting for variable reordering," *Multiple-Valued Logic and Soft Computing*, vol. 13, no. 4-6, pp. 537–552, 2007.

[26] D. M. Miller, D. Y. Feinstein, and M. A. Thornton, "Variable reordering and sifting for QMDD," in *Int'l Symp. on Multi-Valued Logic*, 2007.

[27] J. Watrous, *The theory of quantum information*. Cambridge Univ. Press, 2018.

[28] S. Hillmich, I. L. Markov, and R. Wille, "Just like the real thing: Fast weak simulation of quantum computation," in *Design Automation Conf.*, 2020.

[29] A. Zulehner, P. Niemann, R. Drechsler, and R. Wille, "Accuracy and compactness in decision diagrams for quantum computation," in *Design, Automation and Test in Europe*, 2019, pp. 280–283.

[30] R. Jozsa, "Quantum algorithms and the fourier transform," *Royal Society of London. Series A*, vol. 454, no. 1969, pp. 323–337, 1998.

[31] S. Boixo, S. V. Isakov, V. N. Smelyanskiy, R. Babbush, N. Ding, Z. Jiang, M. J. Bremner, J. M. Martinis, and H. Neven, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, pp. 595–600, 2018.

[32] O. Tange, "GNU parallel: The command-line power tool," *login Usenix Mag.*, vol. 36, no. 1, 2011.

[33] A. Zulehner, S. Hillmich, and R. Wille, "How to efficiently handle complex values? Implementing decision diagrams for quantum computing," in *Int'l Conf. on CAD*, 2019.