Lessons Learnt in the Implementation of Quantum Circuit Simulation Using Decision Diagrams

Thomas Grurl*† Jürgen Fuß* Robert Wille†‡
*Secure Information Systems, University of Applied Sciences Upper Austria, Austria
†Institute for Integrated Circuits, Johannes Kepler University Linz, Austria
‡Software Competence Center Hagenberg GmbH (SCCH), Hagenberg, Austria
{thomas.grurl, juergen.fuss}@fh-hagenberg.at robert.wille@jku.at
http://iic.jku.at/eda/research/quantum/

***Abstract*—Decision diagrams have shown to be a suitable data-structure for tackling the complexity of the quantum world. Accordingly, there has been a lot of research on how to improve their efficiency for quantum circuit simulation as well as broadening their scope. However, there are several smaller yet still interesting aspects that emerge when (re-)implementing corresponding approaches. In this work, we cover these aspects, illustrate them with examples, back them by further experiments, and derive corresponding learnt lessons from these considerations. This eventually gives more detailed insights into the implementation of quantum circuit simulation based on decision diagrams and eventually offers some interesting lessons learnt that may help to use those implementations in a more effective fashion and/or to develop further improvements upon them.**

I. Introduction

Quantum computers promise to solve specific problems significantly faster than classical computers. Early examples thereof are Shor's algorithm for factoring integers [1] and Grover's database search algorithm [2]. More recently, quantum algorithms have also been found for problems in the area of chemistry, finance, machine learning, and mathematics [3]–[6]. In addition to the research on new applications for quantum computers, there have also been tremendous accomplishments towards the physical realization of quantum hardware, which are driven by big players like Google, IBM, Intel, Rigetti, Microsoft, and Alibaba.

Nevertheless, quantum computers are still an emerging technology and current quantum processors are limited in availability and fidelity. Therefore, a considerable amount of research on quantum algorithms still relies on so-called quantum circuit simulators running on classical hardware. From a mathematical point of view, the problem of simulating quantum circuits is simple and boils down to matrix-vector multiplications, with vectors describing quantum states and matrices representing quantum operations. Many state-of-the-art quantum circuit simulators use this concept, representing vectors and matrices as arrays and conduct simulation by matrix-vector multiplication (e.g. [7]–[11]). While conceptually simple, the complexity of this simulation style grows exponentially with the number of simulated qubits—severely limiting those approaches. In order to address this problem of exponentially growing complexity, new approaches for quantum circuit simulation have been developed. A promising approach is based on the use of *decision diagrams* (as introduced in, e.g., [12]–[16]). Decision diagrams offer a compact data-structure for representing and manipulating quantum states, making the approach faster for simulating many types of circuits and/or quantum applications compared to other solutions [17].

Accordingly, several approaches based on decision diagrams have been introduced in the recent past. Besides focusing on an efficient implementation of corresponding simulators [18]–[20], it has also been investigated how to broaden the scope of these approaches. In this context, work has been conducted in the area of equivalence checking of quantum circuits [21], as well as the simulation of quantum circuits with consideration of error effects [22], [23]. While related work as cited above provides detailed descriptions of the respective approaches, there are also several smaller yet still interesting aspects that emerge when (re-)implementing corresponding realizations. These include, e.g., insights into multiplication and addition of decision diagrams, as well as challenges occurring when errors are considered during the simulation that occur in real quantum computers.

In this work, we offer a more detailed coverage of these aspects, that goes beyond what has been discussed in the related work. We distinguish between approaches implementing the "standard" quantum circuit simulation (i.e., assuming error-free quantum systems and basically realizing "pure" matrix-vector multiplication) and approaches additionally considering errors and, hence, realizing a more realistic yet also more complex simulation.

For each aspect, we discuss its potential, illustrate it with examples, back it with further experiments, and derive corresponding learnt lessons from it. By this, we give more detailed insights into the implementation of quantum circuit simulation based on decision diagrams and eventually offer some interesting lessons learnt that may help to use those implementations in a more effective fashion and/or to develop further improvements upon them.

The remainder of this paper is structured as follows: Section II reviews quantum computing and quantum circuit simulation and introduces how decision diagrams can be used for this problem. Section III discusses aspects of multiplying and adding decision diagrams—essential for matrix-vector multiplication, while Section IV addresses challenges specifically arising from simulation of real quantum computers. Finally, Section V concludes the paper.

II. Background

In order to keep this work self-contained, this section briefly introduces quantum computing. We refer the interested reader to [24] for an in-depth introduction.

A. Quantum computing

In the quantum world the basic unit of information is a *quantum bit* or *qubit*. Like a classical bit, qubits can assume the states 0 or 1, which are called basis states and—using Dirac notation are written as $|0\rangle$ and $|1\rangle$. In addition to those basis

states, however, qubits can assume a linear combination of those basis states. More precisely, the state of a qubit $|\psi\rangle$ is described by $\alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle$, with $\alpha_0, \alpha_1 \in \mathbb{C}$. The amplitudes α_0 and α_1 describe how much the qubit is related to each of the basis states. If both amplitudes are non-zero, the qubit is in a so-called *superposition* of both states. Measuring a qubit collapses it to one of the basis states, i.e., the probability of measuring $|0\rangle$ ($|1\rangle$) is given by $|\alpha_0|^2$ ($|\alpha_1|^2$), which can then be observed. The amplitudes themselves are fundamentally unobservable. Naturally, the summed-up probabilities over all basis states must be 1, therefore a valid quantum state of a single qubit must satisfy the normalization constraint $|\alpha_0|^2 + |\alpha_1|^2 = 1$.

These concepts can be extended to describe multi-qubit systems as well. Since each qubit can assume two possible basis states, an n-qubit system can assume 2^n possible basis states and the probability for measuring $|i\rangle$ is given by $|\alpha_i|^2$. Accordingly, the normalization constraint becomes $\sum_{i \in \{0,1\}^n} |\alpha_i|^2 = 1$. Often the state of a qubit is written in the form of a vector of size 2^n containing only the amplitudes. So, e.g., a 2-qubit state would be written like $\begin{bmatrix}\alpha_{00} & \alpha_{01} & \alpha_{10} & \alpha_{11}\end{bmatrix}^\top$.

Quantum states can be manipulated using quantum operations, which are represented by matrices. With exception of the measurement operation—all quantum operations are inherently reversible. They are applied to the state vector using matrix-vector multiplication. Important single-qubit quantum operations are the Hadamard (H) operation, which transforms a basis state into a superposition, the X operation, which negates the state of a qubit, Z operation, which flips the phase of a qubit and the Y, which is a combination of the X and Z operation. These operation are characterized by the matrices $\mathrm{H} = {}^1\!/\!\sqrt{2}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$, $\mathrm{X} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$, $\mathrm{Z} = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$, and $Y = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$. In addition to single-qubit gates, there are also two-qubit gates as well. An important example is the CNOT gate, which flips the state of the target qubit if the first qubit's state is $|1\rangle$.

Example 1. *Consider the two-qubit state $|\psi\rangle$, with both qubits initialized to zero,*

$$1 \cdot |00\rangle + 0 \cdot |01\rangle + 0 \cdot |10\rangle + 0 \cdot |11\rangle\,,$$

which is represented by the vector $\begin{bmatrix}1 & 0 & 0 & 0\end{bmatrix}^\top$.

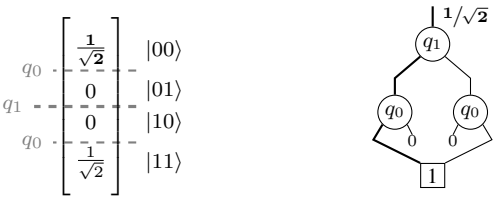
Applying an H operation, which transforms the first qubit from $|0\rangle$ to a superposition and then applying a CNOT to it, which transforms the second qubit of the quantum state if the first qubit is set to $|1\rangle$, is given by

$$\underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix}1\\0\\0\\1\end{bmatrix}}_{|\psi'\rangle} = \underbrace{\begin{bmatrix}1&0&0&0\\0&1&0&0\\0&0&0&1\\0&0&1&0\end{bmatrix}}_{\text{CNOT}} \cdot \underbrace{\frac{1}{\sqrt{2}}\begin{bmatrix}1&0&1&0\\0&1&0&1\\1&0&-1&0\\0&1&0&-1\end{bmatrix}}_{\text{H on the first qubit}} \cdot \underbrace{\begin{bmatrix}1\\0\\0\\0\end{bmatrix}}_{|\psi\rangle}.$$

The state $|\psi'\rangle$ is valid, since $|{}^1\!/\!\sqrt{2}|^2 + 0^2 + 0^2 + |{}^1\!/\!\sqrt{2}|^2 = 1$. Measuring the state yields either $|00\rangle$ or $|11\rangle$, both with probability $|{}^1\!/\!\sqrt{2}|^2 = {}^1\!/\!2$.

B. Quantum Circuit Simulation Using Decision Diagrams

The general idea of decision diagram-based quantum circuit simulation is to exploit structural redundancies within the state vectors representing the quantum state. While in the worst case the size of the decision diagram is still exponential, in many cases they allow to represent states and operations in a very compact form. This in turns allows simulating quantum circuits which cannot be tackled with other simulation approaches.



(a) Vector representation (b) Decision diagram representation

Fig. 1: State vector representation

Decomposing a state vector into a decision diagram revolves around recursively splitting the vector into equally sized sub-vectors, until sub-vectors of size 1, i.e., complex numbers, remain. More precisely, consider a quantum register $q_0, q_1, \ldots, q_{n-1}$ composed of n qubits, where q_{n-1} represents the most significant qubit. The first 2^{n-1} entries of the corresponding state vector represent amplitudes for basis states where q_{n-1} is $|0\rangle$ and the other entries represent amplitudes where q_{n-1} is $|1\rangle$. This is represented in a decision diagram by a node labeled q_{n-1} with two successors labeled q_{n-2}, with the left (right) successor node representing the sub-vector with amplitudes for basis states with q_{n-2} assigned $|0\rangle$ ($|1\rangle$). This process is repeated recursively until complex numbers remain. During the decomposition process identical sub-vectors are represented by the same node and common factors of the amplitudes are stored into the edge weights. Amplitudes can be reconstructed from the decision diagram by multiplying the edge weights along the corresponding path.

Example 2. *Fig. 1 contains the quantum state of $|\psi'\rangle$ in both the vector and decision diagram representation. The annotations in Fig. 1a indicate how the state vector is decomposed into the corresponding decision diagram in Fig. 1b*[1]*. To reconstruct the amplitude of the state $|00\rangle$ the edge weights along the bolded path have to be multiplied, i.e., ${}^1\!/\!\sqrt{2} \cdot 1 \cdot 1 = {}^1\!/\!\sqrt{2}$.*

Matrices—representing quantum operations—are decomposed analogously to vectors. However, due to their square nature, they are split into four equally sized parts, which is represented by a node with four successor nodes.

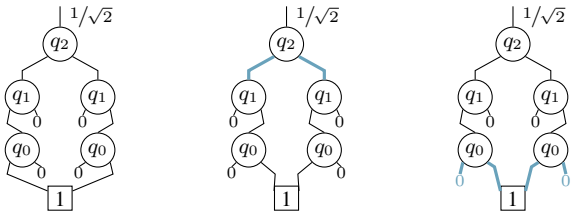
III. Lessons Learnt for Quantum Circuit Simulation

In a first series of considerations, we cover the "standard" quantum circuit simulation using decision diagrams (based on [14]). Recalling the concepts reviewed in Section II, the efficiency of this approach mainly relies on the compact representation of quantum states and operations provided by decision diagrams, as well as a fast realization of matrix-vector multiplications. The underlying basis for the latter is an efficient implementation of multiplication and addition in decision diagrams. Accordingly, we present corresponding lessons learnt on an improved employment of these two operations during quantum circuit simulation.

A. Multiplication on Decision Diagrams

Multiplication is arguably the most important operation for quantum circuit simulation. Hence, it has already been discussed in detail in related work. However, two aspects have not been considered so far: (1) the position of the modified

[1] In order to aid the readability of the decision diagram, edge weights of 1 are omitted. Additionally, nodes with an incoming edge weight of 0 are represented as 0-stubs.



(a) Rep. of $|\phi\rangle$ (b) $|\phi\rangle$ after X to q_2 (c) $|\phi\rangle$ after X to q_0

Fig. 2: Modifications of state $|\phi\rangle$, when applying an X operation

qubit within the decision diagram representation and (2) the type of the applied quantum operation. Both aspects can have significant impact on the cost of applying operations and thus conducting quantum circuit simulation.

To illustrate how the position of the modified qubit affects the cost of applying operations, consider the following example:

Example 3. *Consider the 3-qubit state* $\phi = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}^\top$ *whose decision diagram representation is in Fig. 2a. An* X *operation is applied to the qubit represented by the root node (q_2) and the qubit farthest away from it (q_0):*

- *Applying an* X *operation to q_2 results in swapping the two outgoing edges of the node q_2, as shown in Fig. 2b. Since q_2 is right at the root of the decision diagram, only* one *node has to be visited in order to apply the operation.*
- *Analogously, applying the* X *operation to q_0, requires swapping the outgoing edges of all nodes labeled q_0, resulting in the decision diagram shown in Fig. 2c. However, due to the tree-like structure, accessing the nodes q_0 requires to fully traverse the decision diagram. Moreover, q_0 is represented by two nodes. Thus, applying the operations requires visiting* five *nodes.*

So, although in both cases the same operation is applied, the cost of doing so is very different depending on the target qubit. While only one node has to be visited in order to apply the operation to q_2, modifying q_0 requires to traverse the entire *decision diagram.*

In addition to the position of the modified qubit, the type of operation also affects the cost of applying it. This stems from the fact that many quantum operations, such as $\mathrm{X} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$, $\mathrm{Z} = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$, or $Y = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$, $T = \begin{bmatrix}1 & 0\\ 0 & e^{i\pi/4}\end{bmatrix}$, only modify the target qubit by some factor and/or by swapping of sub-trees. In these cases, the compressed data structure of decision diagrams can be explicitly exploited. More precisely, recall that the amplitudes of the quantum state are encoded within the decision diagram using multiplication (see Section II-B). Thus, modifying a qubit by some factor does not require decompressing any amplitudes, but can instead be done by directly modifying the nodes representing the specific qubit. The cost of applying such operations therefore becomes the cost of accessing all nodes representing the qubit the operation is applied to. Since decision diagrams encode the quantum state in a tree-like structure, qubits closer to the root can be easily accessed—making applying such operations potentially extremely fast.

Unfortunately, this effect cannot be exploited for all quantum operations. Examples of operations where this effect cannot be exploited are $\mathrm{H} = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$, $\mathrm{R}_x(\frac{\pi}{2}) = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -i\\ -i & 1\end{bmatrix}$ or $\mathrm{R}_y(\frac{\pi}{2}) = \frac{1}{\sqrt{2}}\begin{bmatrix}1 & -1\\ 1 & 1\end{bmatrix}$. Applying those operations not only requires accessing the nodes representing the target qubit, but also access to all sub-nodes. Thus, it is not relevant how deep the target qubit is within the decision diagram, since it has to be traversed anyway.

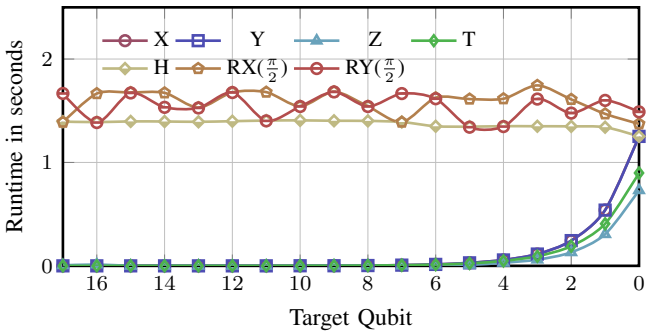


Fig. 3: Average executing time for applying operation depending on the target qubit

Example 4. *To illustrate this effect, we tracked the average time it took to apply specific quantum operations to an 18-qubit quantum system, depending on the target qubit. We applied the operation always to the same quantum state, which we prepared in a way to contain little redundancies, using Google's supremacy circuit [25]. The results are shown in Fig. 3. The Y-axis show the average execution time of the operation, while the X-axis indicates how deep the target qubit is within the decision diagram, with 17 denoting that the target qubit is directly represented by the root node and 0 denoting that the target qubit is at the bottom. The figure clearly shows that, when applying an H, $\mathrm{R}_x(\frac{\pi}{2})$, or $\mathrm{R}_y(\frac{\pi}{2})$ operation, the cost is unaffected by position of the target qubit. In contrast, when an X, Z, Y, or T operation is applied, the cost depends on how deep the target qubit is within the decision diagram.*

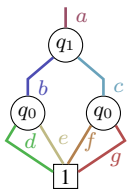
Lessons Learnt:

- The position of the target qubit within the decision diagram is important. If the target qubit is represented by the root node, potentially only this node has to be accessed. The farther away the representation of the target qubit is from the root, the more nodes need to be considered.
- The type of operation has an impact on whether the position of the target qubit matters. If, e.g., an H, $\mathrm{R}_x(\frac{\pi}{2})$, or $\mathrm{R}_y(\frac{\pi}{2})$ operation is applied, the position of the target qubit does not matter at all. For other operations (such as X, Z, Y and T), it has an impact.

B. Addition on Decision Diagrams

Multiplication is indisputably the most important operation during quantum circuit simulation. Addition, however, is also important, due to it being a necessary subroutine of matrix-vector multiplication. This makes adding decision diagrams an operation worth considering. In fact, adding decision diagrams is not straightforward as the following example shows:

Example 5. *Adding two decision diagrams is similar to adding vectors (or matrices), i.e., all values sharing the same index have to be accessed and summed up. This is problematic for decision diagrams since the values are encoded within the tree structure. Fig. 4 illustrates how a generic 2-qubit state vector is decomposed into a decision diagram (see Fig. 4a) and how the individual edge weights relate to the amplitudes of the*



$$\begin{bmatrix} a \cdot b \cdot d \\ a \cdot b \cdot e \\ a \cdot c \cdot f \\ a \cdot c \cdot g \end{bmatrix}$$

(a) Generic 2-qubit DD (b) Generic 2-qubit state vector

Fig. 4: Decomposition of amplitudes within a decision diagram (DD)

corresponding state vector (see Fig. 4b). Adding two decision diagrams therefore becomes a full recursive traversal of them both, so that all amplitudes can be restored and summed up. Additionally, after adding the amplitudes, they have to be again decomposed into the decision diagram representation.

Naturally, restoring all amplitudes of the decision diagram is slow. Other decision diagram decomposition schemes are possible, which allow adding decision diagrams without having to decompress the stored values. However, doing so without making the multiplication slower is problematic. The decision diagram decomposition presented in Section II-B is optimized for the more important multiplication operation. Thus, operations—which only require multiplying decision diagrams—can be potentially applied extremely fast (as illustrated in Section III-A).

Nevertheless, two shortcuts are possible compared to the naive approach described in Example 5: Firstly, when a zero edge is encountered during the recursive traversal in one of the decision diagrams, all amplitudes along this path are zero. Therefore, the new amplitudes are simply the ones stored in the remaining decision diagram. Secondly, when two identical (sub-)edges are added, one edge can instead be multiplied by a factor of two. Due to the compressed data structure, identical values are represented by the same node within decision diagrams, which makes finding duplicate paths straightforward.

Lessons Learnt:

- Adding decision diagrams cannot be conducted directly on the data structure but instead requires restoring the decomposed elements. Making the operation rather inefficient and slow in general.
- Improving the performance of the addition without making the multiplication less efficient is difficult. Nevertheless, improvements are possible by exploiting zero amplitudes as well as duplicate entries.

IV. Lessons Learnt for Quantum Circuit Simulation with Consideration of Errors

Due to the fragile nature of quantum systems, today's quantum architectures are plagued by frequent unavoidable errors [26]. Considering those errors during quantum circuit simulation makes a hard problem even harder and requires a dedicated approach. Nevertheless, decision diagrams have proven to be a promising tool to handle this additional complexity [22], [23]. But also here, aspects exist from which further lessons learnt can be retrieved. Those are covered in this section. To this end, we consider two types of corresponding simulation approaches: deterministic and stochastic. In the following, both types are first reviewed before the aspects leading to the corresponding lessons learnt are discussed.

A. Deterministic Consideration of Errors

Errors in quantum computing are understood and quantum mechanics is capable of describing them. However, considering them during quantum circuit simulation in a deterministic fashion requires extending the models described in Section II. First of all, in order to describe a quantum state effected by errors, the state description must be extended from vectors of size 2^n to *density matrices* (also known as *density operators*) of size $2^n \times 2^n$. This new density matrix ρ is derived from the state vector $|\psi\rangle$ description by

$$\rho = |\psi\rangle \langle\psi| \text{ with } \langle\psi| := |\psi\rangle^\dagger . \tag{1}$$

Applying an operation U to such a quantum representation ρ now consist of two matrix-matrix multiplications in the form of $U\rho U^\dagger$. Despite the additional complexity, the concepts for efficient multiplications and addition of decision diagrams can be reused for matrix-matrix multiplications. Unfortunately, this does not fully extend to applying error effects.

Since errors are probabilistic, i.e., they occur by chance, their representation and application differs compared to "intentional" operations. More precisely, using the operator-sum representation, an error is described by a tuple $(E_0, E_1, \ldots, E_m)$ of *Kraus matrices* that satisfy the condition

$$\sum_{i=0}^{m} E_i^\dagger E_i = I \tag{2}$$

and are applied to a quantum state (represented by a density matrix ρ) by

$$\rho' = \sum_{i=0}^{m} E_i \rho E_i^\dagger . \tag{3}$$

Example 6. *Suppose that an error could occur to $|\psi'\rangle$ from Example 2 that dampens a qubit from a high energy state $|1\rangle$ to $|0\rangle$ (in the literature, this is usually called a T1 error) [24]. To capture this, the density matrix ρ is generated from the vector description using Eq. (1)*[2]*,*

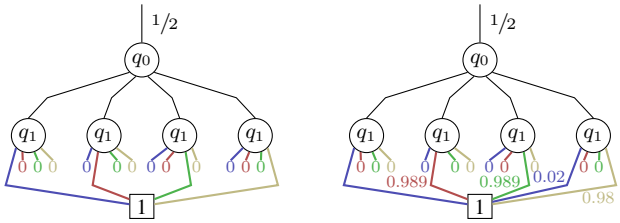
$$\underbrace{\begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}}_{\rho} = \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix}}_{|\psi'\rangle} \cdot \underbrace{\begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix}}_{\langle\psi'|} .$$

The T1 error can be described by the Kraus matrices $E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}$ and $E_1 = \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix}$. Assume that the error affects the second qubit with a probability of 2 % ($p = 0.02$). Applying both Kraus matrices to the state ρ (as defined in Eq. (3)) leads to

$$\underbrace{\begin{bmatrix} 0.5 & 0 & 0 & 0.494 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0.494 & 0 & 0 & 0.49 \end{bmatrix}}_{\rho'} = \underbrace{\begin{bmatrix} 0.5 & 0 & 0 & 0.494 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.494 & 0 & 0 & 0.49 \end{bmatrix}}_{(I \otimes E_0)\rho(I \otimes E_0)^\dagger} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{(I \otimes E_1)\rho(I \otimes E_1)^\dagger} .$$

The resulting density matrix ρ' accordingly describes the effect of the error: While the probability for measuring $|00\rangle$ remains the same, the probability of measuring $|11\rangle$ has dropped to 49 % and, additionally, there is now a probability of 1 % of measuring $|10\rangle$.

[2]The probabilities for measuring specific basis states are reflected within the diagonal of ρ.



(a) DD rep. of ρ' (b) ρ' after T1 error with $p = 0.02$

Fig. 5: Effects of applying an error to a decision diagram (DD)

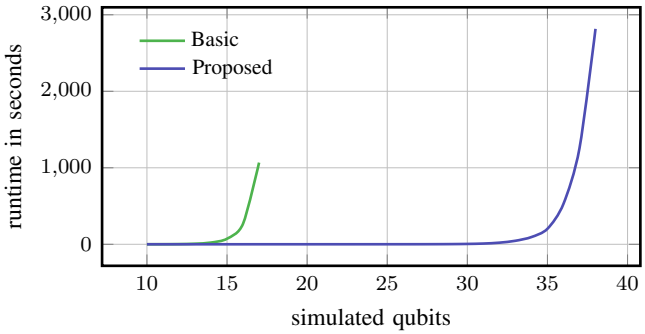


Fig. 6: Runtime for simulating the quantum Fourier transform with consideration of decoherence errors

Thus, applying errors effects comes down to realizing Eq. (3). However, naively implementing this scheme reduces the performance considerably, because (1) each Kraus matrix is individually applied to the quantum state, (2) adding fully sized decision diagrams is slow (see Section III-B), and (3) often more than one error effect is being considered, so that this slow procedure has to be repeated for each considered error.

By exploiting the decision diagram structure, this can be done in a more efficient way. To illustrate this consider the following example:

Example 7. *Consider again Example 6, where a T1 error affects qubit q_1 of the state ρ with 2 % probability. However, now we exploit the fact that every qubit is represented in a decision diagram by one or more nodes. Modifying each node representing a specific qubit corresponds to modifying the qubit itself. Therefore, we first calculate how the T1 error affects a single qubit,*

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \longmapsto \begin{bmatrix} a + dp & \sqrt{1-p} \cdot b \\ \sqrt{1-p} \cdot c & (1-p) \cdot d \end{bmatrix}. \tag{4}$$

Next, we modify all nodes labeled q_1 according to the error effect (the color illustrates how each matrix value corresponds to the edges in the decision diagram), leading to Fig. 5b. This new decision diagram represents ρ after the T1 decoherence error has been applied and is equal to ρ' from Example 6.

Applying errors in such a fashion can considerably reduce the execution time. To illustrate this, we conducted runs where we considered decoherence errors for simulating the quantum Fourier transform with increasing number of qubits. In Fig. 6 the results are plotted. The Y-axis gives the runtime, while in the X-axis the number of simulated qubits is given. The plot clearly shows the performance improvements compared to the basic approach.

Lessons Learnt:

- Complex operations consisting of several steps (e.g., Eq. (3)) can often be merged and, by this the number of times individual elements have to be accessed, can be considerably reduced—improving the performance for decision diagram-based quantum circuit simulation.

B. Stochastic Simulation of Errors

Stochastic quantum circuit simulation allows consideration of errors without the exponential overhead that comes with deterministic simulation. This comes with the drawback however, that the deterministic description is lost. When using this approach, simulation is conducted as presented in Section II, i.e., states are represented by 1-dimensional vectors and operations are applied by matrix-vector multiplications. Error effects are viewed as unwanted operations that randomly affect the state. Therefore, whenever an error might affect the quantum state it is randomly chosen if the effect is applied or not based on its probability of occurrence.

Example 8. *Consider again the 2-qubit state $|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ from Example 1. Suppose that the first qubit of this state might be affected by an error which depolarizes it (i.e., setting it into a completely random state) with 1 % probability. We can capture the effect of depolarization by either applying I[3], X, Y, or Z—each with probability $\frac{p}{4}$, where p is the probability that the error is applied [24]. Therefore, after the error might have occurred, with 99.25 % the state remains unchanged and it becomes either $\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle)$, $\frac{i}{\sqrt{2}}(-|01\rangle + |10\rangle)$, $\frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)$ each with 0.25 % probability.*

After simulating in such a fashion, one possible final state is generated. This opens the door to Monte Carlo sampling. By repeating this process, a sufficient number of times and averaging the probabilities, the actual final state can be approximated.

Therefore, the stochastic quantum circuit simulation approach can be directly implemented on top of the concepts from Section II. However, by exploiting the compressed data structure of decision diagrams, this process can be optimized by exploiting an idea first suggested in [27]. To illustrate this idea, consider again that quantum circuit simulation comes down to matrix-vector multiplications. More precisely, m quantum operations $O_0, O_1, \ldots, O_{m-1}$ are applied to an initial state $|\psi\rangle$, resulting in the final state $|\psi\rangle^m$, i.e.,

$$|\psi\rangle^m = O_{m-1} \cdot O_{m-2} \ldots O_0 \cdot |\psi\rangle \tag{5}$$

Since matrix-vector multiplications are generally cheaper than matrix-matrix multiplications, Eq. (5) is usually solved from right to left. When using decision diagrams, however, this does not necessarily hold true. Since decision diagrams exploit redundancies in the data structure, the decision diagram representation of quantum operations can be considerably more compact than decision diagrams representing complex states. Thus, stacking (i.e. multiplying) quantum operations with each other before they are applied to the quantum state can be potentially faster.

The viability of this approach depends on finding a good heuristic of how many operations are stacked before they are applied to the quantum state. Additionally, it also depends on

[3]The Identity (I) operation leaves a state unchanged and is an important concept when simulating with errors.

TABLE I: Experimental results

Benchmark	#Qubits	Basic [s]	Proposed [s]	Improv.
basis_trotter	4	62.40	47.75	23.48 %
error_cor.	5	8.02	5.99	25.31 %
vqe_uccsd	6	142.33	87.65	38.42 %
qaoa	6	11.29	8.95	20.73 %
ising	10	377.78	279.30	26.07 %
cc	12	24.01	21.94	8.62 %
qf21	15	7.60	5.57	26.71 %
bv	19	215.92	169.96	21.29 %

the size of the quantum state, since when the decision diagram representation of the quantum state is compact, stacking operations is not efficient.

Both aspects are addressed, when this approach is used for stochastic error simulation, i.e., whenever an "intentional" quantum operation is applied, the error affects are stacked onto it. Firstly, this is a good heuristic for stacking the operations, because all operations target the same qubit, keeping the resulting stacked operation, compact. Secondly, considering error effects during the simulation of quantum circuits automatically destroys redundancies in the state, resulting in a more complex state description and increasing the size of the decision diagram representing it.

To evaluate the potential of this strategy we took the circuits from the benchmark suite *QASMBench* (taken from [28]) and simulated them with consideration of errors. We simulated the circuits with a "basic" implementation, where operations are only applied sequentially, as well as with the proposed solution. In Table I, the results are summarized. Note that, due to space limitations, experiments whose runtimes do not differ significantly are omitted. The results clearly demonstrate the potential of this strategy.

Lessons Learnt:

- When working with decision diagrams operations on elements with higher dimensions (e.g., matrices) can be actually cheaper than on more complex elements of lower dimensions (e.g., vectors).

V. Conclusion

Decision diagrams are a promising tool for tackling the complexity of the quantum world. Several decision diagram-based approaches have already been introduced, addressing more efficient implementations as well as investigating new applications. However, there are several smaller yet still interesting aspects that emerge when (re-)implementing corresponding approaches. This work sheds light on those aspects and derives corresponding lessons learnt. By this a better understanding of decision diagrams for quantum computing is reached and potential for future work has been unveiled.

Acknowledgments

This work has partially been supported by the University of Applied Sciences PhD program of the State of Upper Austria (managed by the FFG), by the LIT Secure and Correct Systems Lab funded by the State of Upper Austria, as well as by the BMK, BMDW, and the State of Upper Austria in the frame of the COMET program (managed by the FFG). Furthermore, this project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 101001318).

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] Y. Cao, J. Romero, J. P. Olson, M. Degroote, P. D. Johnson, M. Kieferová, I. D. Kivlichan, T. Menke, B. Peropadre, N. P. Sawaya *et al.*, "Quantum chemistry in the age of quantum computing," *Chemical reviews*, vol. 119, no. 19, pp. 10 856–10 915, 2019.

[4] P. Rebentrost, B. Gupt, and T. R. Bromley, "Quantum computational finance: Monte Carlo pricing of financial derivatives," *Phys. Rev. A*, vol. 98, 2018.

[5] I. Kerenidis, J. Landman, A. Luongo, and A. Prakash, "q-means: A quantum algorithm for unsupervised machine learning," *Proc. of the Neural Information Processing Systems*, 2019.

[6] E. Farhi, J. Goldstone, and S. Gutmann, "A quantum approximate optimization algorithm," *arXiv:1411.4028*, 2014.

[7] M. Smelyanskiy, N. P. D. Sawaya, and A. Aspuru-Guzik, "qHiPSTER: The quantum high performance software testing environment," *Computing Research Repository*, vol. abs/1601.07195, 2016.

[8] G. Aleksandrowicz *et al.*, "Qiskit: An Open-source Framework for Quantum Computing," *Zenodo*, 2019.

[9] D. Steiger, T. Häner, and M. Troyer, "ProjectQ: An open source software framework for quantum computing," *Quantum*, vol. 2, 2018.

[10] "Cirq: A python framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuits." https://github.com/quantumlib/Cirqgithub.com/quantumlib/Cirq, 2019, accessed: 2020-01-22.

[11] Atos SE, "Quantum learning machine," https://atos.net/en/products/quantum-learning-machineatos.net/en/products/quantum-learning-machine, 2016, Accessed: 2019-11-20.

[12] V. Samoladas, "Improved BDD algorithms for the simulation of quantum circuits," in *European Symp. on Algorithms*, 2008, pp. 720–731.

[13] G. Viamontes, I. Markov, and J. Hayes, "High-performance QuIDD-based simulation of quantum circuits," in *Design, Automation and Test in Europe*, 2004, pp. 1354–1355.

[14] A. Zulehner and R. Wille, "Advanced simulation of quantum computations," 2018.

[15] D. M. Miller, M. A. Thornton, and D. Goodman, "A decision diagram package for reversible and quantum circuit simulation," in *IEEE World Congress on Computational Intelligence*, 2006, pp. 8597–8604.

[16] 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.

[17] T. Grurl, J. Fuß, S. Hillmich, L. Burgholzer, and R. Wille, "Arrays vs. decision diagrams: A case study on quantum circuit simulators," in *Int'l Symp. on Multi-Valued Logic*, vol. 50, 2020, pp. 176–181.

[18] A. Zulehner and R. Wille, "Matrix-Vector vs. Matrix-Matrix Multiplication: Potential in DD-based Simulation of Quantum Computations," in *Design, Automation and Test in Europe*, 2019, pp. 90–95.

[19] P. Niemann, A. Zulehner, R. Drechsler, and R. Wille, "Overcoming the trade-off between accuracy and compactness in decision diagrams for quantum computation," in *IEEE Trans. on CAD of Integrated Circuits and Systems*. IEEE, 2020.

[20] 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.

[21] L. Burgholzer and R. Wille, "Advanced equivalence checking for quantum circuits," *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2021.

[22] T. Grurl, J. Fuß, and R. Wille, "Considering decoherence errors in the simulation of quantum circuits using decision diagrams," in *Int'l Conf. on CAD*, 2020, pp. 1–7.

[23] T. Grurl, R. Kueng, J. Fuß, and R. Wille, "Stochastic quantum circuit simulation using decision diagrams," in *Design, Automation and Test in Europe*, 2021.

[24] M. Nielsen and I. Chuang, *Quantum Computation and Quantum Information*. Cambridge Univ. Press, 2000.

[25] S. Boixo, S. V. Isakov, V. N. Smelyanskiy *et al.*, "Characterizing quantum supremacy in near-term devices," *Nature Physics*, vol. 14, no. 6, pp. 595–600, 2018.

[26] J. Preskill, "Quantum Computing in the NISQ era and beyond," *Quantum*, vol. 2, p. 79, Aug. 2018. [Online]. Available: https://doi.org/10.22331/q-2018-08-06-79

[27] A. Zulehner and R. Wille, "Matrix-vector vs. matrix-matrix multiplication: Potential in DD-based simulation of quantum computations," in *Design, Automation and Test in Europe*, 2019.

[28] A. Li and S. Krishnamoorthy, "QASMBench: A Low-level QASM Benchmark Suite for NISQ Evaluation and Simulation," 2020. [Online]. Available: http://arxiv.org/abs/2005.13018