Considering Decoherence Errors in the Simulation of Quantum Circuits 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

By using quantum mechanical effects, quantum computers promise significant speedups in solving problems intractable for conventional computers. However, despite recent progress they remain limited in scaling and availability—making quantum software and hardware development heavily reliant on quantum simulators running on conventional hardware. However, most of those simulators mimic *perfect* quantum computers and, hence, ignore the fragile nature of quantum mechanical effects which frequently yield to decoherence errors in real quantum devices. Considering those errors during the simulation is complex, but necessary in order to tailor quantum algorithms for specific devices. Thus far, most state-of-the-art simulators considering decoherence errors rely on (exponentially) large array representations. As an alternative, simulators based on decision diagrams have been shown very promising for simulation of quantum circuits in general, but have not supported decoherence errors yet. In this work, we are closing this gap. We investigate how the consideration of decoherence errors affects the simulation performance of approaches based on decision diagrams and propose advanced solutions to mitigate negative effects. Experiments confirm that this yields improvements of several orders of magnitudes compared to a naive consideration of errors.

1 INTRODUCTION

Quantum computers can solve specific problems significantly faster than classical computers. Besides early examples of corresponding algorithms such as Shor's factorization approach [23] and Grover's database search [8], recently also other relevant quantum algorithms have been found in the areas of chemistry, finance, machine learning, and mathematics [6, 11, 13, 17, 21]. Moreover, not least due to the drive of big companies such as IBM, Google, Intel, Rigetti, Microsoft, or Alibaba, who are heavily investing in this emerging technology, there have been remarkable accomplishments towards the physical realization of quantum computers recently.

However, quantum computers do not work perfectly and are indeed even more prone to errors than their classical counterparts. In particular, decoherence errors [26] are a common phenomenon with which researchers and engineers frequently have to deal with. They lead to the problem that qubits can only hold information for a limited amount of time. Although recent developments in the physical realization of quantum computers have improved upon that [5, 12], decoherence errors are are still a dominating aspect in quantum computing.

Accordingly, it is essential to specifically evaluate whether quantum algorithms can cope with this kind of errors. To this end, methods for quantum circuit simulation would be ideal as they allow for an explicit investigation how a given algorithm behaves on a specific device and its possible errors. But most of the available simulators (such as, e.g., proposed in [16, 19, 25, 28, 34]) mimic *perfect* quantum computers and, thus, ignore the fragile nature of quantum mechanical effects which frequently yield to decoherence errors. Luckily, the effects are well understood and mathematical models for decoherence errors are available [18]—leading to first simulation approaches to also support the consideration of those errors (e.g. [1–3, 10, 14, 24, 29, 30]).

But a main challenge remains: The capabilities of those simulation approaches is severely limited by the inherent exponential nature of the vectors and matrices which describe the respective quantum states and operations, respectively. Additionally considering decoherence errors further increases the resulting complexity. Because of this, alternatives to straightforward approaches (which represent the required vectors and matrices in terms of arrays) are currently investigated. Decision diagrams for quantum circuit simulation (as introduced, e.g., in [16, 19, 22, 27, 34]) provide a promising approach. In many cases and/or for many quantum applications, they allow for a representation of vectors and matrices which is below the exponential size of array-based solutions [9, 34].

All these methods, however, do not support the consideration of decoherence errors yet. Moreover, thus far, it remains unknown whether the promising effects of a more efficient representation of vectors and matrices can still be maintained, when those errors are additionally taken into account. In this work, we are investigating this issue. Our observations and evaluations suggest that decision diagrams might remain compact in many cases—even if decoherence errors are considered. But we also show that just having a compact representation is not sufficient: Efficiently realizing the operations describing the error effects poses a substantial challenge. In order to mitigate those negative effects, we eventually propose

an advanced solution for simulation using decision diagrams that also can efficiently handle decoherence errors.

Experimental evaluations and comparisons to state-of-the-art simulators by IBM and Atos confirm the viability of the proposed solution. Moreover, we show that the proposed advanced solution is able to complete simulation runs several orders of magnitudes faster than the solution with a naive consideration of errors.

Our contributions are described in the rest of this paper as follows: Section 2 reviews quantum computing and quantum circuit simulation using arrays as well as decision diagrams. Section 3 discusses how considering errors changes the way simulation is conducted and analyzes the resulting challenges for approaches based on decision diagrams. Based on these insights, we propose an advanced simulation scheme in Section 4. Finally, we summarize our evaluations in Section 5 and conclude the paper in Section 6.

2 BACKGROUND

In this section, we review the basics of quantum computing and the simulation of corresponding quantum algorithms/circuits (using array-based methods as well as methods based on decision diagrams).

2.1 Quantum Computing

While, in the classical world, systems are described by bits which can either be 0 or 1, in quantum computing a system is described by so-called quantum bits or qubits. In contrast to classical bits, qubits can assume not only the state 0 or 1—which are called basis states and, using Dirac notation, are written as $|0\rangle$ and $|1\rangle$—but also an almost arbitrary combination (*superposition*) of these states. More precisely, the state of the qubit $|\phi\rangle$ is given by $|\phi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle$. The amplitudes $\alpha_0, \alpha_1 \in \mathbb{C}$ describe how the qubit is related to each of the basis state and must satisfy the normalization condition $|\alpha_0|^2 + |\alpha_1|^2 = 1$. Measuring a qubit in superposition results in its collapse into one of the basis states $|0\rangle$ or $|1\rangle$ with probability $|\alpha_0|^2$ and $|\alpha_1|^2$, respectively.

These concepts can be extended for multi-qubit systems—to so-called *quantum registers*—to represent the exponential number of basis states the system can assume. For example, a two qubit system $|\psi\rangle$ has four basis states and is described by $|\psi\rangle = \alpha_{00} \cdot |00\rangle + \alpha_{01} \cdot |01\rangle + \alpha_{10} \cdot |10\rangle + \alpha_{11} \cdot |11\rangle$. Usually, the state description for n-qubit systems is shortened to a column vector of size 2^n containing only the amplitudes, e.g., $[\alpha_{00}\ \alpha_{01}\ \alpha_{10}\ \alpha_{11}]^\top$ for $n = 2$.

The state of a qubit can be manipulated using quantum operations. With the exception of the measurement operation, all quantum operations are inherently reversible and, therefore, represented by unitary matrices. Important quantum operations are the NOT operation, which negates the state of a qubit, and the Hadamard operation, which transforms a qubit from a basis state into a superposition. In addition to single-qubit operations, there are two-qubit operations. A prominent operation is the controlled-NOT (CNOT) operation, which negates the state of a qubit, if the designated control qubit is in state $|1\rangle$. Applying an operation to a quantum state is done via matrix-vector multiplication. To illustrate these concepts, consider the following example:

Example 1. *Consider a two-qubit register in the state*

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

which is represented as $1/\sqrt{2} \cdot [1\ 0\ 1\ 0]^\top$. *This is a valid state since* $|1/\sqrt{2}|^2 + 0^2 + |1/\sqrt{2}|^2 + 0^2 = 1$. *Measuring the state yields either* $|00\rangle$ *or* $|10\rangle$, *both with probability* $|1/\sqrt{2}|^2 = 1/2$. *Applying a CNOT operation to the state, which flips the amplitude of the second qubit when the first qubit, is set to 1 is given by*

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

Measuring the new state $|\psi'\rangle$ *now yields either* $|00\rangle$ *or* $|11\rangle$, *each with probability* $1/2$.

2.2 Array-based Quantum Circuit Simulation

The array-based simulation style realizes the concepts described above in a straightforward fashion. States and operations are represented by 1-dimensional and 2-dimensional arrays, respectively. Simulation is conducted by matrix-vector multiplications similar to Example 1. Since the multiplication of a matrix and a vector can be decomposed into smaller operations, this simulation approach has a huge potential for parallelization. Every matrix-vector multiplication can be split into a series of multiplications and additions, i.e.,

$$\begin{bmatrix} M_{00} & M_{01} \\ M_{10} & M_{11} \end{bmatrix} \cdot \begin{bmatrix} V_0 \\ V_1 \end{bmatrix} = \begin{bmatrix} M_{00} \cdot V_0 + M_{01} \cdot V_1 \\ M_{10} \cdot V_0 + M_{11} \cdot V_1 \end{bmatrix}.$$

This can be further decomposed in a recursive fashion, leading to a large set of intermediate operations, which can be executed independently of each other with little synchronization overhead. State-of-the-art array-based simulators (such as [1–3, 14, 24, 25, 30]) make heavy use of that.

2.3 Decision Diagram-based Quantum Circuit Simulation

In order to tackle the memory problem of array-based simulators, a complementary approach has been developed using decision diagrams [16, 19, 22, 27, 34]. The general idea of decision diagram-based simulation is about identifying data redundancies in the state and representing them using shared sub-structures. Doing so results in a potentially very compact state representation, which in turn allows simulating quantum applications that cannot be simulated using array-based simulation approaches.

Representing a state vector as a decision diagram revolves around recursively splitting the vector into equal sized sub-vectors, until the sub-vectors only contain a single element. More precisely, consider a quantum register $q_0, q_1, \ldots, q_{n-1}$ composed of n qubits, where q_0 represents the most significant qubit. The first 2^{n-1} entries of the corresponding state vector would then represent amplitudes for basis states where q_0 is $|0\rangle$ and the other entries would represent amplitudes where q_0 is $|1\rangle$. This is represented in a decision
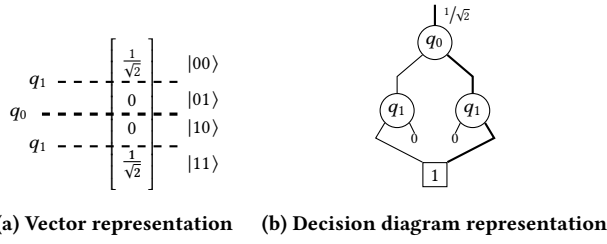
(a) Vector representation   **(b) Decision diagram representation**

Figure 1: State vector representation

diagram by a node labeled q_0 with two successors labeled q_1. By convention, the left (right) successor points to a node that represents the sub-vector with amplitudes for basis states with q_0 assigned $|0\rangle$ ($|1\rangle$). This process is repeated recursively until sub-vectors of size 1 (i.e. complex numbers) result. During this process, equivalent sub-vectors are represented by the same node, reducing the overall size of the decision diagram. Furthermore, instead of having distinct terminal nodes for all amplitudes, edge weights are used to store common factors of the amplitudes, leading to even more compaction. Reconstructing the amplitude of a specific state can be done by multiplying the edge weights along the corresponding path. In order to increase 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—indicating that amplitudes of all possible states represented by this part of the decision diagram are zero.

Example 2. *In Fig. 1, a quantum register is represented both in the vector and decision diagram representation. The annotations of the state vector in Fig. 1a indicates how it is decomposed when the corresponding decision diagram is constructed. Reconstructing the amplitude for a specific state from the decision diagram requires multiplying the edge weights of the corresponding path. For example, reconstructing the amplitude of the state $|11\rangle$ (bold lines in the figure), requires multiplying the edge weight of the root edge ($^1/\sqrt{2}$) with the right edge of q_0 (1) as well as q_1 (1), i.e. $^1/\sqrt{2} \cdot 1 \cdot 1 = {}^1/\sqrt{2}$.*

Quantum operations are represented in a similar fashion as quantum states. However, due to the square nature of matrices, they are split into four equal sized sub-parts. This is represented in a decision diagram by a node with four successors the first one representing the sub-matrix in the upper left corner, the second one representing the sub-matrix in the upper right corner, the third one representing the sub-matrix in the lower left corner, and the fourth one representing the sub-matrix in the lower right corner. Apart from this, the decomposition and normalization process is analogue to the one for vectors.

Similar to array-based simulators, simulation is conducted by multiplying operations onto states. However, due to the different representation, the multiplication must be decomposed with respect to the most significant qubit. Consider again a quantum register $|\phi\rangle = q_0, q_1, \ldots, q_{n-1}$ of n qubits, where q_0 represents the most significant qubit, as well as a unitary quantum operation U of size $2^n \times 2^n$. In order to multiply the operation U onto the state $|\phi\rangle$, they are split into two (in the case of the state vector) and four (in the case of the operation) equally sized parts. Leading to two sub-vectors of size 2^{n-1} and four sub-matrices each of size $2^{n-1} \times 2^{n-1}$. This represents the modifications of U onto q_0 and is represented by a top

node labeled q_0, with two successor nodes. The splitting process is repeated recursively until vectors of size 2 and matrices of size 2×2 remain. These are multiplied and the resulting new amplitudes are stored into terminal edges. Finally, the edge weights are calculated by extracting common factors of the amplitudes and equivalent sub-vectors are represented by the same node. Hence, multiplication of decision diagrams mainly involves recursive traversals of the involved decision diagrams. On top of that, further optimizations are possible with respect to the precision of the simulation [20, 33], the run-time performance [35], or a trade-off of both [31]

3 SIMULATION OF DECOHERENCE

In this work, we aim for the consideration of decoherence errors in the simulation of quantum circuits using decision diagrams. To this end, we provide the motivation and propose an initial approach in this section. More precisely, this section first reviews decoherence effects that occur in today's quantum computer realizations and afterwards provides a mathematical description of them to be used by simulation approaches in general. Based on that, we then show how those descriptions can be used to conduct such simulations using decision diagrams and discuss how this may affect the performance of corresponding approaches.

3.1 Qubit Decoherence Errors

Coherence errors occur due to the fragile nature of qubits—leading to the problem that they can only hold information for a limited amount of time. More precisely, there are two types of coherence errors that arise [26]:

- A qubit in a high-energy state $|1\rangle$ naturally tends to decay to a low energy state $|0\rangle$, i.e., after a certain amount of time, qubits in a quantum system eventually relax to $|0\rangle$. This error is called *amplitude damping error* or *T1 error*.
- In addition to that, when a qubit interacts with the environment, a phase flip effect might occur. This leads to an error called *phase flip error* or *T2 error*.

Recent developments in the physical realization of quantum computers (such as, e.g., in [5, 12]) show significant improvements in the coherence times and improved possibilities to reduce unwanted interactions of qubits—improving the "lifetime" of qubits before decaying to $|0\rangle$ and reducing the frequency of phase flip errors, respectively. Nevertheless, the underlying errors are still a dominating aspect in all quantum computations and, hence, should also be considered during simulation.

However, in order to describe them mathematically, the formulation in terms of state vectors as reviewed in Section 2.1 obviously is not sufficient. Moreover, even a deterministic formulation is not suitable since decoherence heavily relies on probabilistic effects that are not known in advance. Hence, a description is needed which incorporates all possible states a quantum system may be in (including the original state but also states resulting from any of the decoherence effects mentioned above with a certain probability). This is accomplished by extending the vector representation introduced in Section 2 to *density matrices* (also known as *density operators*) as follows:

Definition 1. *Let $|\phi\rangle$ be a complex vector representing the state of a quantum system. The corresponding* density matrix *is defined as*

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

Example 3. *Consider a system, which is in the state*

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

or, using vector notation, $\frac{1}{\sqrt{2}} \cdot [1\ 0\ 0\ 1]^{\top}$. The corresponding density matrix ρ *is given by*

$$\begin{bmatrix} \frac{1}{\sqrt{2}} \\ 0 \\ 0 \\ \frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & 0 & \frac{1}{\sqrt{2}} \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 & 0 & \frac{1}{2} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & \frac{1}{2} \end{bmatrix}.$$

In contrast to the original vector representation (as illustrated in Example 1), the probabilities of measuring specific basic states are now reflected in the diagonal elements (highlighted in gray). More precisely, the diagonal entries from the first element in the upper-left to the last element in the bottom-right represent the probabilities for measuring $|00\rangle$, $|01\rangle$, $|10\rangle$, and $|11\rangle$, respectively. Hence, measuring this state would yield $|00\rangle$ or $|11\rangle$—both with probability of $\frac{1}{2}$. All other elements of the density matrix represent the coherence in the state.

Employing this concept allows to probabilistically apply decoherence, i.e., the application (or non-application) of decoherence effects by means of probabilities. More precisely, the effects of a decoherence error can be described by *Kraus matrices* defined as follows:

Definition 2. *Using the operator-sum representation, an error is represented by a tuple $(E_0, E_1, \ldots, E_m)$ of* Kraus matrices *that satisfy the condition*

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

Using this notation, the T1 and T2 errors can be represented by [18]

$$T1 = (E_0, E_1) \ with \ E_0 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}, E_1 = \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix} \ and \tag{3}$$

$$T2 = (E_0, E_1) \ with \ E_0 = \sqrt{p} \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, E_1 = \sqrt{1-p} \cdot \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{4}$$

respectively, where the variable p represents the probability that an error occurs. This probability is a parameter of the specific quantum computer realization (and, hence, needs to be provided by the user).

Applying these error descriptions to the current quantum system (represented by a density matrix) can be conducted as follows:

Definition 3. *Applying an error specified by the Kraus matrices $(E_0, E_1, \ldots, E_m)$ to a quantum system given by the density matrix ρ yields the density matrix [18]*

$$\rho' = \sum_{i=0}^{m} E_i \rho E_i^{\dagger}. \tag{5}$$

Example 4. *In order to illustrate the concepts above, we apply the amplitude damping (T1) error to the state ρ from Example 3. More precisely, we apply amplitude damping to the second qubit with a probability of 30 % (p=0.3). The effects of this error are given by the Kraus matrices provided in Eq. 3. Applying each Kraus matrix to the state ρ (as defined in Eq. 5) leads to*

$$\underbrace{\begin{bmatrix} 0.5 & 0 & 0 & 0.418 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0.418 & 0 & 0 & 0.35 \end{bmatrix}}_{E_0 \rho E_0^{\dagger}} + \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.15 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{E_1 \rho E_1^{\dagger}} = \underbrace{\begin{bmatrix} 0.5 & 0 & 0 & 0.418 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0.15 & 0 \\ 0.418 & 0 & 0 & 0.35 \end{bmatrix}}_{\rho'}.$$

The resulting density matrix accordingly describes the effect of the employed error: While the probability for measuring $|00\rangle$ remains the same, the probability of measuring $|11\rangle$ has dropped to 35 % and, additionally, there is now a probability of 15 % to measure $|10\rangle$. In other words, the probability that the second qubit is measured $|0\rangle$ has increased by 30 %—reflecting the damping error assumed above.

Finally, since states are now represented by (density) matrices, applying an operation to a state has to be adjusted. Originally, given a quantum state $|\phi\rangle$ and a quantum operation U, the corresponding application is conducted by matrix-vector multiplication (as illustrated in Example 1). Now, vector states $|\phi\rangle$ are represented by density matrices given by $\rho = |\phi\rangle\langle\phi|$ (cf. Definition 1 and Eq. 1). Accordingly, applying U onto ρ is given by

$$\rho' = U \rho U^{\dagger}. \tag{6}$$

3.2 Effect to Simulation

Having the mathematical description of decoherence errors as reviewed above, simulation approaches can accordingly be extended. This is particularly straightforward for array-based approaches as proposed in [1–3, 14, 24, 25, 30] and reviewed in Section 2.2. Here, the major challenge of considering decoherence is "only" given by the fact that the state is represented by density matrices rather than vectors. This can simply be addressed by extending the corresponding data structures (i.e., the arrays). The required operations (in particular, for multiplying and adding matrices) are supported by the corresponding libraries anyway and can also be extended in a straightforward fashion. Also, improvements employed by parallel executions can readily be utilized. Because of that, implementations of array-based simulation approaches supporting the consideration of decoherence are already available (see, e.g., [1–3, 10, 14, 24, 30]).

For approaches based on decision diagrams, the consideration of decoherence errors, however, may pose much more severe challenges (which have not be considered in the literature thus far). First, it remains open how severely the consideration of decoherence errors (and the corresponding need of representing states and operations through density matrices) harms the ability of decision diagrams to represent states and operations in a compact fashion. As reviewed in Section 2.3, being able to represent certain quantum states and quantum operations in a more compact fashion constitutes one of the main advantages of simulation using decision diagrams. The question remains whether this compactness can still be maintained when decoherence errors are additionally considered. To this end, consider the following example:
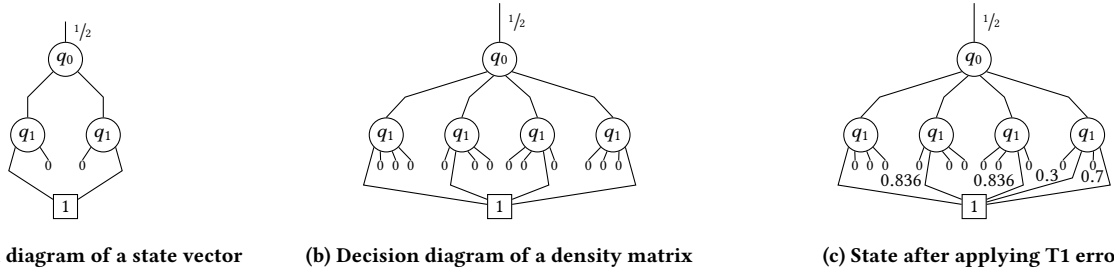
(a) Decision diagram of a state vector

(b) Decision diagram of a density matrix

(c) State after applying T1 error

Figure 2: Decision diagram representation of states

EXAMPLE 5. *Recall the state considered before in Example 3 in both the vector as well as in the density matrix representation. The corresponding decision diagram representations are provided in Fig. 2a and Fig. 2b, respectively. As can be seen, the size of the decision diagrams are rather similar (4 nodes vs. 6 nodes, although the corresponding vector/density matrix have a size of 2^2 and $2^2 \times 2^2$). This, at least, shows that considering decoherence errors does not necessarily harm the ability of decision diagrams to represent states/operations in a compact fashion.[1] Moreover, also more detailed evaluations (summarized later in Section 5) confirm this observation.*

Considering the example (and the evaluations summarized in Section 5) suggests that decision diagrams might remain compact in many cases—even if decoherence errors are considered. But having a compact representation alone is not sufficient. Also efficient realizations of the corresponding matrix-matrix operations (most notably multiplication and addition), as reviewed in Section 3.1, are required. For multiplication, related work such as [34] already provides efficient solutions since matrix-matrix multiplication already is a core operation of simulation without decoherence. Addition, however, which is required for applying the error effect as defined in Eq. 5, has not been that frequently required thus far and turns out to be particularly challenging.

More precisely, recall that adding two matrices is done by adding all elements sharing the same index. Hence, this requires access to *all* single matrix elements—something for which a decision diagram has to be *completely* traversed. This is in stark contrast with multiplication, where addition is required as well, but the operands are often just sub-parts of the involved decision diagrams (which substantially reduces the size of the decision diagrams that need to be traversed). Additionally, decision diagram representations of operations are of very sparse nature, so that during the multiplication often one part of the addition equals zero.

Overall, this suggests that decision diagrams also show promise for quantum circuit simulation when considering decoherence errors (something which has not been considered thus far). At the same time, it also unveils challenges which remains to be addressed, namely how to efficiently apply the decoherence to the state without having to traverse the entire decision diagram.

---

[1]Note that this does not mean that decision diagrams *always* provide a compact representation for a quantum state/operation. In fact, previous work [34] clearly showed that the worst-case complexity of decision diagrams is exponential—even though polynomial representations are possible for certain applications. The example, however, shows that this characteristic is not completely lost when considering decoherence errors.

4 ADVANCED SIMULATION APPROACH

In order to address the shortcomings unveiled above, we investigated how the required operations (particularly Eq. 5) can efficiently be realized on decision diagrams. A major obstacle is that applying decoherence effects heavily relies on the addition of matrices which requires access to *all* single matrix elements and, in turn, triggers the complete traversal of the decision diagrams—causing an exponential overhead independently of how compact they can be represented.

Accordingly, we looked for alternatives that either completely avoid the addition of matrices or, at least, only conduct it on (smaller) sub-matrices. Our investigations eventually resulted in such alternatives whose main idea rests on the following three observations:

First, adding two matrices is not always necessary. In particular, the T2 error can be described by multiplications only. In fact, w.l.o.g., the effect of this error on a single qubit can be described by

$$\begin{bmatrix} a & b \\ c & d \end{bmatrix} \longmapsto \begin{bmatrix} a & (2p-1)b \\ (2p-1)c & d \end{bmatrix},$$

where p represents the probability. That is, two of the elements are not changed at all, while the remaining ones are just multiplied by $2p - 1$. Hence, applying the T2 error can be reduced to two multiplications—without the need for any addition.

Second, applying the error as specified in Eq. 5 is inefficient and allows to apply only one error effect to one qubit at a time. By explicitly enforcing the error effects directly on the corresponding nodes of the affected qubits, we can apply all desired effects to all qubits with just one traversal of the decision diagram.

Third, if the matrix is represented in terms of decision diagrams, it is not always necessary to do the addition on the entire matrix. In fact, in decision diagrams, every qubit is represented by one or more nodes. Applying a decoherence error to a qubit actually only modifies the outgoing edges from the nodes representing this specific qubit. All predecessor nodes are only indirectly affected when the edge weights are normalized (as described in Section 2.3).

Based on those observations, we are proposing an advanced method of applying this operation on decision diagrams: Instead of doing the matrix-matrix multiplications followed by the addition separately, we apply the decoherence effects in one step—exploiting our observation that applying an operation to a qubit only modifies the outgoing edge weights of nodes representing it. Decoherence can therefore accordingly be applied by directly modifying all those edges. In doing so, we also gain full control over how the operation is applied—allowing us to exploit the first two observations as well.

EXAMPLE 6. *The advanced approach is illustrated by reusing Example 4. Recall that, in this example, we apply an amplitude damping (T1) decoherence error with a probability of 30 % to q_1 of the quantum state ρ from Example 3. In contrast to the earlier example, we now apply the error directly to the decision diagram of ρ (given in Fig. 2b). To this end, we apply the effects of the T1 error to all nodes labeled q_1 leading to the following transformations*

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

As defined in Section 2.3, the four outgoing edges of a node in a decision diagram represent the matrix elements a, b, c and d from left to right. Hence, modifying the fourth node labeled q_1 with probability $p = 0.3$ leads to

$$\begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix} \longmapsto \begin{bmatrix} 0 + 1 \cdot 0.3 & 0 \\ 0 & 1 - 1 \cdot 0.3 \end{bmatrix} = \begin{bmatrix} 0.3 & 0 \\ 0 & 0.7 \end{bmatrix}$$

The other nodes labeled q_1 are modified in the same way, leading to the new decision diagram shown in Figure 2c. This new decision diagram represents ρ after the T1 decoherence error has been applied and is equal to ρ' from Example 4.

5 EVALUATION

In this section, we summarize the core results of our evaluations conducted in order to investigate the effect of considering decoherence errors in the simulation of quantum circuits using decision diagrams. To this end, we took the state-of-the-art decision diagram-based simulator from [32, 34] and extended this implementation to additionally support decoherence errors. This led to one version in which error support has been added in a straightforward fashion (i.e., directly applying the concepts described in Section 3.1) and another version in which error support has been added in an advanced fashion (i.e., applying the methods described in Section 4). In addition to that and for the purpose of comparison, we also considered two state-of-the-art array-based simulators, namely *Linalg* from the commercial *Atos Quantum Learning Machine* (QLM) [3] as well as the *density_matrix* simulator from IBM's Qiskit [2].

For the evaluations, we assumed the amplitude damping (T1) error with 0.2 % probability and the phase flip (T2) error with 0.1 % probability (applied each time a qubit has been used). As quantum algorithm, we considered the *Quantum Fourier Transform* (QFT [18]) with an increasing number of qubits. This is an ideal choice for the purpose of this evaluation, since QFT (1) constitutes a core element of numerous existing quantum applications (such as Shor's factorization method [23], quantum phase estimation [18] and the hidden subgroup problem [18]), (2) is an established benchmark for evaluating the effects of decoherence errors in related work (e.g., [4]), and (3) additionally has the benefit that, thus far, it represents one of the best cases for simulation using decision diagrams without considering decoherence (showing a linear scalability in memory and run-time with respect to the number of qubits). Therefore, the effects of considering decoherence errors can ideally be investigated using QFT.

All evaluations have been conducted on a system using 5 cores running at a clock frequency of 2.2 GHz and 1.5 TB of RAM. While the QLM simulator runs directly on this system, we ported the Qiskit simulator and the simulators using decision diagrams to this machine utilizing Docker [15]. We choose Docker since its virtualization overhead is negligible [7]. By this, all simulators have been evaluated using the same hardware resources.

Table 1 summarizes the obtained results. The first column provides the number of considered qubits. In the remaining columns, we list the peak memory usage in MB as well as the total simulation time in real time seconds for all considered approaches, i.e., Atos' and IBM's array-based QLM and Qiskit, respectively, as well as the considered decision diagrams-based approach with and without the improvement from Section 4. Note that the accumulated CPU time of the array-based approaches would be substantially larger than the real time values listed in Table 1, since both approaches heavily utilize concurrency enabled by the available five cores during the simulation. Cells without any entries represent instances where the timeout of one hour has been exceeded.

The results clearly confirm the observations from above: Considering decoherence errors during simulation does not necessarily harm the compact representation of decision diagrams. In fact, the respective memory requirements for the decision diagrams remain rather moderate (never more than 150 MB), while they sky-rocket for the array-based approaches (certainly, the main reason why those approaches only scale up to 15 or 14 qubits, respectively). At the same time, it also can be seen that run-time becomes much more the limiting factor for the decision diagram-based simulation. This confirms the discussion from Section 3.2 and shows the impact of the advanced method of conducting the required operations as described in Section 4—eventually yielding improvements of several orders of magnitude and more than twice the scalability.

6 CONCLUSION

Decision diagrams provide a promising alternative for quantum circuit simulation due to their capability of representing vectors and matrices in a much more compact fashion than, e.g., array-based methods. But no work existed yet which investigated whether these promising effects can be maintained when decoherence errors—still, a dominating aspect in quantum computing—are additionally assumed. This work sheds light on this. We observed that considering decoherence errors not necessarily harms the compact representation, but leads to new challenges with respect to conducting the respectively required operations in a run-time-efficient fashion. In order to address these challenges an advanced method has been proposed, which mitigates the negative effects and led to improvements of several orders of magnitudes. By this, we showed that quantum circuit simulation using decision diagrams remains a promising approach also when decoherence errors are considered.

Table 1: Experimental results

| | Array-based | | | | Decision diagram-based | | | |
| | QLM | | Qiskit | | Without Improvement | | With Improvement | |
#Qubits	Mem [MB]	Time [s]	Mem [MB]	Time [s]	Mem [MB]	Time [s]	Mem [MB]	Time [s]
10	142.11	2.06	261.57	40.04	94.44	0.47	41.94	0.16
11	190.21	4.29	313.65	97.27	95.03	0.87	41.00	0.16
12	390.43	13.54	505.40	174.69	97.62	2.40	39.97	0.17
13	1166.96	58.22	1297.79	366.26	98.22	7.47	40.48	0.17
14	4241.86	265.40	4287.89	1114.23	100.58	22.49	40.74	0.17
15	16535.59	1217.52			103.10	72.89	43.21	0.17
16					107.69	269.04	41.70	0.17
17					109.55	1067.78	94.46	0.27
18							97.21	0.27
						⋮		⋮
36							129.85	528.62
37							135.08	1255.23
38							137.02	2816.20

REFERENCES

[1] 2019. Cirq: A Python Framework for Creating, Editing, and Invoking Noisy Intermediate Scale Quantum (NISQ) Circuits. github.com/quantumlib/Cirq. Accessed: 2020-01-22.

[2] Héctor Abraham et al. 2019. Qiskit: An Open-source Framework for Quantum Computing. https://doi.org/10.5281/zenodo.2562110

[3] Atos SE. 2016. Quantum Learning Machine. atos.net/en/products/quantum-learning-machine. Accessed: 2019-11-20.

[4] Himanshu Chaudhary, Biplab Mahato, Lakshya Priyadarshi, Naman Roshan, Azad Utkarsh, and Apoorva Patel. 2019. A Software Simulator for Noisy Quantum Circuits. arXiv:1908.05154 (2019).

[5] Michael Devoret and Robert Schoelkopf. 2013. Superconducting Circuits for Quantum Information: An Outlook. Science 339 (2013), 1169–1174.

[6] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. 2014. A Quantum Approximate Optimization Algorithm. arXiv:1411.4028 (2014).

[7] Wes Felter, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. 2015. An updated performance comparison of virtual machines and Linux containers. In 2015 IEEE International Symposium on Performance Analysis of Systems and Software. 171–172. https://doi.org/10.1109/ISPASS.2015.7095802

[8] Lov K. Grover. 1996. A Fast Quantum Mechanical Algorithm for Database Search. In Symp. on Theory of Computing. 212–219. https://doi.org/10.1145/237814.237866

[9] Thomas Grurl, Jürgen Fuß, Stefan Hillmich, Lukas Burgholzer, and Robert Wille. 2020. Arrays vs. Decision Diagrams: A Case Study on Quantum Circuit Simulators. In Int'l Symp. on Multi-Valued Logic, Vol. 50. 176–181.

[10] Tyson Jones, Anna Brown, Ian Bush, and Simon Benjamin. 2018. QuEST and High Performance Simulation of Quantum Computers. arXiv preprint arXiv:1802.08032 (2018).

[11] Ivan Kassal, Stephen Jordan, Peter Love, Masoud Mohseni, and Alán Aspuru-Guzik. 2008. Polynomial-time quantum algorithm for the simulation of chemical dynamics. Proc. of the National Academy of Sciences 105, 48 (2008), 18681–18686.

[12] Julian Kelly. 2018. A Preview of Bristlecone, Google's New Quantum Processor. https://ai.googleblog.com/2018/03/a-preview-of-bristlecone-googles-new.html Accessed: 2019-05-19.

[13] Iordanis Kerenidis, Jonas Landman, Alessandro Luongo, and Anupam Prakash. 2019. q-means: A quantum algorithm for unsupervised machine learning. Proc. of the Neural Information Processing Systems (2019).

[14] Nader Khammassi, Imran Ashraf, Xiang Fu, Carmen Almudever, and Koen Bertels. 2017. QX: A High-Performance Quantum Computer Simulation Platform. In Design, Automation and Test in Europe.

[15] Dirk Merkel. 2014. Docker: Lightweight Linux Containers for Consistent Development and Deployment. Linux Jour. 2014, 239, Article 2 (2014).

[16] D. M. Miller, M. A. Thornton, and D. Goodman. 2006. A Decision Diagram Package for Reversible and Quantum Circuit Simulation. In IEEE World Congress on Computational Intelligence. 8597–8604.

[17] Ashley Montanaro. 2016. Quantum algorithms: An overview. npj Quantum Information 2 (2016), 15023.

[18] Michael Nielsen and Isaac Chuang. 2000. Quantum Computation and Quantum Information. Cambridge Univ. Press.

[19] Philipp Niemann, Robert Wille, David Michael Miller, Mitchell A. Thornton, and Rolf Drechsler. 2016. QMDDs: Efficient Quantum Function Representation and Manipulation. 35, 1 (2016), 86–99.

[20] Philipp Niemann, Alwin Zulehner, Rolf Drechsler, and Robert Wille. 2020. 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.

[21] Patrick Rebentrost, Brajesh Gupt, and Thomas R. Bromley. 2018. Quantum computational finance: Monte Carlo pricing of financial derivatives. Phys. Rev. A 98 (2018), 15. Issue 2. https://doi.org/10.1103/PhysRevA.98.022321

[22] Vasilis Samoladas. 2008. Improved BDD Algorithms for the Simulation of Quantum Circuits. In European Symp. on Algorithms. 720–731.

[23] Peter W. Shor. 1997. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. 26, 5 (1997), 1484–1509.

[24] Mikhail Smelyanskiy, Nicolas P. D. Sawaya, and Alán Aspuru-Guzik. 2016. qHiP-STER: The Quantum High Performance Software Testing Environment. Computing Research Repository abs/1601.07195 (2016).

[25] Damian Steiger, Thomas Häner, and Matthias Troyer. 2018. ProjectQ: An open source software framework for quantum computing. Quantum 2 (2018).

[26] Swamit Tannu and Moinuddin Qureshi. 2018. Not All Qubits Are Created Equal. arXiv:1805.10224 (2018).

[27] George Viamontes, Igor Markov, and John Hayes. 2004. High-performance QuIDD-based simulation of quantum circuits. In Design, Automation and Test in Europe. 1354–1355.

[28] Guifré Vidal. 2003. Efficient classical simulation of slightly entangled quantum computations. Physical review letters 91, 14 (2003).

[29] Benjamin Villalonga, Sergio Boixo, Bron Nelson, et al. 2019. A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware. npj Quantum Information 5, 1 (2019).

[30] Dave Wecker and Krysta Svore. 2014. LIQUi|>: A Software Design Architecture and Domain-Specific Language for Quantum Computing. arXiv:1402.4467 (2014).

[31] Alwin Zulehner, Stefan Hillmich, Igor Markov, and Robert Wille. 2020. Approximation of Quantum States Using Decision Diagrams. In Asia and South Pacific Design Automation Conf. 121–126.

[32] Alwin Zulehner, Stefan Hillmich, and Robert Wille. 2019. How to Efficiently Handle Complex Values? Implementing Decision Diagrams for Quantum Computing. In Int'l Conf. on CAD.

[33] Alwin Zulehner, Philipp Niemann, Rolf Drechsler, and Robert Wille. 2019. Accuracy and Compactness in Decision Diagrams for Quantum Computation. In Design, Automation and Test in Europe. 280–283.

[34] Alwin Zulehner and Robert Wille. 2018. Advanced Simulation of Quantum Computations. IEEE Trans. on CAD of Integrated Circuits and Systems.

[35] Alwin Zulehner and Robert Wille. 2019. Matrix-Vector vs. Matrix-Matrix Multiplication: Potential in DD-based Simulation of Quantum Computations. In Design, Automation and Test in Europe. 90–95.