Stochastic Quantum Circuit Simulation Using Decision Diagrams

Thomas Grurl*† Richard Kueng† 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 {richard.kueng, robert.wille}@jku.at
http://iic.jku.at/eda/research/quantum/

***Abstract*—Recent years have seen unprecedented advance in the design and control of quantum computers. Nonetheless, their applicability is still restricted and access remains expensive. Therefore, a substantial amount of quantum algorithms research still relies on simulating quantum circuits on classical hardware. However, due to the sheer complexity of simulating real quantum computers, many simulators unrealistically simplify the problem and instead simulate *perfect* quantum hardware, i.e., they do not consider errors caused by the fragile nature of quantum systems. *Stochastic quantum simulation* provides a conceptually suitable solution to this problem: physically motivated errors are applied in a probabilistic fashion throughout the simulation. In this work, we propose to use decision diagrams, as well as concurrent executions, to substantially reduce resource-requirements—which are still daunting—for stochastic quantum circuit simulation. Backed up by rigorous theory, empirical studies show that this approach allows for a substantially faster and much more scalable simulation for certain quantum circuits.**

I. Introduction

By utilizing quantum mechanical effects, quantum computers promise to solve problems which are intractable for classical computers. Early examples for this are Shor's algorithm [1] for factoring integers or Grover's database search algorithm [2]. As the research on quantum algorithms gained more and more traction, more quantum algorithms have been found in the areas of chemistry, finance, machine learning, and mathematics [3]–[6]. Alongside the work of quantum software development, there have been unprecedented accomplishments towards the physical realization of quantum hardware. In 2019, Google claimed to have achieved quantum advantage by using a 54-qubit processor to calculate a task in 200 seconds for which they estimate a state-of-the-art supercomputer would require ten thousand years [7]. In the same year, IBM released its 53-qubit quantum computer and made it available for commercial use [8]. And, recently, IBM announced their roadmap towards launching a 1212-qubit processor in 2023 [9].

However, recent breakthroughs notwithstanding, quantum computers are still an emerging technology and current quantum processors are limited in reliability and availability. Consequently, a considerable amount of research on quantum algorithms still relies on simulating quantum circuits on classical hardware. This is an exponentially hard task, almost by definition. To make matters worse, classical simulation of a *perfect* quantum circuit is, arguably, beside the point. Today's quantum architectures are plagued by frequent errors that are unavoidable given the fragile nature of quantum systems [10]. Although error mitigation is constantly improving, they are still a dominating factor in quantum computing. Therefore, taking those errors into account when simulating quantum circuits is essential in order to understand how an algorithm behaves when executed on real quantum hardware.

As a theory, quantum mechanics is capable of describing these types of errors and their effect (namely quantum channels and mixed states [2]). However, this general formalism renders the exponentially hard problem of quantum circuit simulation even harder—accordingly limiting corresponding simulation approaches (see, e.g., [11]–[20]). This is why many quantum circuit simulators simplify the problem and only mimic *perfect* (i.e., error-free) quantum computers (e.g., [21]–[25]).

In this work, we consider an alternative approach which avoids making a hard problem unnecessarily harder. Instead, we consider a stochastic error model. That is, we assume that errors occur randomly throughout individual simulation runs. Afterwards, we approximate the true effect of errors on a quantum computation by forming empirical averages over multiple simulation runs (Monte Carlo). This provides a conceptually suitable and mathematically rigorous solution for classical simulation of noisy quantum computations which can easily be implemented on top of existing simulators, such as [11]–[13].

Nonetheless, severe challenges remain. First and foremost, there is the curse of dimensionality: performing a single simulation run requires repeated matrix-vector multiplications of exponential size to appropriately track the effect of quantum operations. To make matters worse, a single stochastic simulation run does not capture error effects appropriately. Instead, a sufficiently large number of independent simulation runs must be conducted to form empirical averages that accurately reflect the true quantum evolution. Both factors combined render existing solutions severely limited with respect to efficiency and scalability.

In order to overcome these limitations, we propose a solution which (1) uses decision diagrams to represent states and operations in a more compact fashion and (2) conducts concurrent executions to accelerate the generation of samples. Evaluations and comparisons to state-of-the-art simulators by IBM and Atos confirm the viability of the proposed solution. In fact, for certain circuits, we were able to conduct the respective simulations in a more scalable fashion (i.e,. considering substantially more qubits than before) and a much more efficient fashion (often, several orders of magnitudes faster).

Our contributions are described in the rest of this paper as follows: Section II reviews quantum computing and the errors that might occur. Section III discusses how those errors can be simulated using a stochastic approach. In Section IV, we outline the concept of the proposed solution, which we then evaluate in Section V against two state-of-the-art simulators. Finally, Section VI concludes the paper.

II. Background

In order to keep this work self-contained, this section reviews the basic concepts of quantum computing as well as error effects. We refer the interested reader to standard textbooks, e.g., [2], [26], for a more thorough introduction.

978-3-9819263-5-4/DATE21/©2021 EDAA

A. Quantum Computing

In the classical world, the basic unit of information is a bit, which can either assume the state 0 or 1. In the quantum world, the smallest unit of information is called a *quantum bit* or *qubit*. Like a classical bit, a qubit can assume the states 0 and 1, which are called *basis states* and—using Dirac notation—are written as $|0\rangle$ and $|1\rangle$. Additionally, a qubit can also assume an almost arbitrary combination of the two basis states, which is then called a *superposition*. More precisely, the state of the qubit $|\psi\rangle$ is written as $|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle$ with $\alpha_0, \alpha_1 \in \mathbb{C}$ such that $|\alpha_0|^2 + |\alpha_1|^2 = 1$. The values α_0, α_1 are called *amplitudes* and describe how strongly the qubit is related to each of the basis states. Measuring a qubit yields $|0\rangle$ ($|1\rangle$) with probability $|\alpha_0|^2$ ($|\alpha_1|^2$). By measuring the qubit, any existing superposition is destroyed and the state of the qubit collapses to the measured basis state.

Quantum states containing more than one qubit are often called *quantum registers* and the concepts above can be extended to describe such systems as well. An n qubit register can assume $N = 2^n$ basis states and is described by N amplitudes $\alpha_0, \alpha_1, \ldots \alpha_{N-1}$, which must satisfy the normalization constraint $\sum_{i \in \{0,1\}^n} |\alpha_i|^2 = 1$. Usually quantum states are shortened to state vectors containing only the amplitudes, e.g., $\begin{bmatrix}\alpha_{00} & \alpha_{01} & \alpha_{10} & \alpha_{11}\end{bmatrix}^\top$ for $n = 2$ qubits.

Example 1. *Consider the 2-qubit quantum register*

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

which is represented by the state vector $\begin{bmatrix}\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0\end{bmatrix}^\top$. *This is a valid state, since it satisfies the normalization constraint* $\left|\frac{1}{\sqrt{2}}\right|^2 + 0^2 + \left|\frac{1}{\sqrt{2}}\right|^2 + 0^2 = 1$. *Measuring the system yields either* $|00\rangle$ *or* $|10\rangle$*—both with probability* $\left|\frac{1}{\sqrt{2}}\right|^2 = \frac{1}{2}$. *Note that the leftmost qubit is in a superposition and equally strongly related to* $|0\rangle$ *and* $|1\rangle$*, while the other qubit is in the basis state* $|0\rangle$.

Quantum states can be manipulated using quantum operations. With the exception of the measurement operation, all quantum operations are inherently reversible and represented by unitary matrices, i.e., square matrices whose inverse is their conjugate transpose. Important 1-qubit operations are $\mathrm{H} = 1/\sqrt{2}\begin{bmatrix}1 & 1\\ 1 & -1\end{bmatrix}$ (transforming a basis state into a superposition), $\mathrm{X} = \begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}$ (the quantum equivalent of the NOT operation), $\mathrm{Z} = \begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}$ (flipping the phase of a qubit), as well as the combination of both $Y = iXZ = \begin{bmatrix}0 & -i\\ i & 0\end{bmatrix}$. Another "operation" is the identity operation given by $\mathrm{I} = \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}$. It simply leaves a state unchanged and is relevant in the context of simulating errors. There are also 2-qubit operations. An important example is the controlled-X (also known as CNOT) operation, which negates the state of a qubit, if the chosen control qubit is $|1\rangle$. Applying an operation to a state can be done by matrix-vector multiplication.

Example 2. *Consider again the 2-qubit register* $|\psi\rangle$ *from Example 1. Applying a CNOT operation to* $|\psi\rangle$*, which negates the amplitude of the second qubit if the first qubit is set to* $|1\rangle$*, is given by*

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

Measuring $|\psi'\rangle$ *either yields* $|00\rangle$ *or* $|11\rangle$*, each with probability* $1/2$. *Note that the measurement outcome of one qubit affects the other one as well—an essential concept in quantum computing known as* entanglement.

B. Errors in Quantum Computing

The formalism presented above can be used to describe how perfect quantum computers behave. However, due to the fragile nature of quantum systems, real quantum computers are prone to errors. These errors can be classified into two categories [27]: *Gate errors* (also known as operational errors) and *coherence errors* (also known as retention errors).

1) Gate Errors are introduced by executed operations [27]. They occur since quantum computers are mechanical constructions that do not always apply operations perfectly. Instead, the operation may be not executed at all, or in a (slightly) modified fashion. Since gate errors are highly specific for each quantum computer and even vary for qubits within the quantum computer, they are often approximated using depolarization errors [12], [14]. The depolarization error describes that a qubit is set to a completely random state [2]. For publicly available quantum computers from IBM, the error probabilities are in the order of 10^{-3} to 10^{-2} [28].

2) Coherence Errors occur due to the fragile nature of quantum systems (qubits). In practice, this leads to the problem that they can hold information for a limited time only. There are two types of coherence errors that may appear [27]:

- A qubit in a high-energy state ($|1\rangle$) tends to relax into a low energy state ($|0\rangle$). That is, after a certain amount of time, qubits in a quantum system eventually decay 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*.

Developments in the physical realization of quantum computers (e.g., in [29], [30]) show significant improvements in the coherence times of qubits—improving the "lifetime" of qubits before decaying to $|0\rangle$ and reducing the frequency of phase flip errors, respectively. Nevertheless, the errors are still a significant aspect in all quantum computations and, hence, should also be considered during simulation.

Error effects can be viewed as (unwanted) operations on the state. However, while ideal quantum operations are deterministic, errors can have an additional degree of randomness. For instance, a 1-qubit amplitude damping channel may fire (with probability p) or it may do nothing (with probability $1 - p$). The outcome of an erroneous quantum computation cannot be described as a single state $|\psi\rangle$ anymore. Instead, it is described by an *ensemble* of possible outcomes $\{(p_i, |\psi_i\rangle)\}$. Here, the states $|\psi_i\rangle$ label potential outcomes while each weight p_i describes the probability with which outcome $|\psi_i\rangle$ occurs ($p_i \geq 0$ and $\sum_i p_i = 1$).

Example 3. *Consider again the 2-qubit state* $|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ *from Example 2. Suppose that this state might be affected by a gate error in the first qubit only, depolarizing it. With probability* $1 - p$*, nothing happens and the state remains unchanged. With probability* p*, the first qubit becomes depolarized. We can capture this effect by either applying I, X, Y, or Z—each with probability* $\frac{p}{4}$. *This produces an ensemble (or mixture)* $\{(1-p, \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), (\frac{p}{4}, \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle), (\frac{p}{4}, (\frac{1}{\sqrt{2}}(|01\rangle + |10\rangle), (\frac{p}{4}, (\frac{i}{\sqrt{2}}(-|01\rangle + |10\rangle), (\frac{p}{4}, \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle)\}$ *which cannot be represented by a single 2-qubit state.*

III. Stochastic Quantum Circuit Simulation

In order to conduct quantum circuit simulation, the concepts described in Section II need to be emulated on a classical machine. Conceptually, this can be conducted in a straightforward fashion: State vectors and operation matrices are first represented in the form of 1-dimensional and 2-dimensional arrays, respectively. Then, the application of quantum states is handled by applying matrix-vector multiplication as illustrated in Example 2 above.

However, the problem of this approach is that the representation of both, quantum states and quantum operations require exponentially large vectors and matrices—rendering quantum circuit simulation very complex. Moreover, error effects occur only by chance, i.e., they are randomly applied depending on the hardware model of the simulated quantum computer and cannot simply be considered in a pre-defined fashion (such as operations). Therefore, instead of one possible final state, simulation with errors produces a range of possible output states, depending on the applied error effects (as illustrated in Example 3 above).

This phenomenon is well known and may be captured by a rigorous mathematical formalism: quantum channels and mixed states [2], [26]. However, this formalism necessarily amplifies the curse of dimensionality: mixed states correspond to $2^n \times 2^n$ matrices and keeping track of them renders an exponentially large problem even harder.

Stochastic quantum simulation, on the other hand, avoids this further increase in simulation complexity by sacrificing deterministic descriptions. The key idea is to imitate error effects in a real quantum computer. That is, whenever the quantum computer might make an error during its calculation with some probability p, we mimic the effect of this error during the simulation with probability p and leave the state untouched with probability $1-p$. By simulating in such a way, we generate *one possible* final state $|\tilde{\psi}\rangle$ that is sampled from the actual ensemble $\{(p_i, |\psi_i\rangle)\}$: $|\tilde{\psi}\rangle = |\psi_i\rangle$ with probability p_i. Sampling access opens the door for stochastic (Monte-Carlo) approximation: simply approximate the true distribution by forming empirical averages of sampled output states.

Stochastic approximation is particularly well suited for directly and accurately learning interesting properties of the final state (distribution) without the need of keeping track of the complete distribution. In quantum computing, many interesting properties can be described in terms of quadratic functions in the state vector, i.e., $o_l = |\langle \omega_l | \psi \rangle|^2$. Prominent examples are the fidelity with another state, as well as the outcome probability of a computational basis measurement. For a probabilistic state mixture $\{(p_i, |\psi_i\rangle)\}$ such a quadratic property becomes

$$o_l = \sum_i p_i \left| \langle \omega_l | \psi_i \rangle \right|^2 \tag{1}$$

and can be approximated by an empirical average over M samples $|\tilde{\psi}_j\rangle$ from this distribution:

$$\hat{o}_l = \tfrac{1}{M} \sum_{j=1}^{M} \left| \langle \omega_l | \tilde{\psi}_j \rangle \right|^2 \quad \text{(Monte Carlo)}.$$

Moreover, the same collection of samples $\left\{ |\tilde{\psi}_1\rangle, \ldots, |\psi_M\rangle \right\}$ can be used to estimate many quadratic properties at once.

Theorem 1. *Fix a collection of L (arbitrary) quadratic properties* (1)*, as well as $\epsilon \in (0,1)$ (accuracy) and $\delta \in (0,1)$ (confidence). Then, $M = \log(2L/\delta)/(2\epsilon)^2$ state samples suffice to accurately approximate* all *target properties with high confidence: $\max_l |\hat{o}_l - o_l| \leq \epsilon$ with probability at least $1-\delta$.*

Proof. Fix a target property $o_l = \sum_i p_i |\langle \omega_l | \psi_i \rangle|^2$. Conducting a single stochastic run yields the correct property in expectation, i.e., $\mathbb{E}|\langle \omega_l | \tilde{\psi}_j \rangle|^2 = o_l$. Standard concentration inequalities, like Hoeffding, imply $\Pr\left[|o_l - \hat{o}_l| \geq \epsilon\right] \leq 2\mathrm{e}^{-2M\epsilon^2}$. The claim follows from taking a union bound over all L target approximations and inserting the advertised value of M. □

As is typical of Monte Carlo, the required number of samples M scales inverse quadratically in the desired accuracy ϵ. More importantly and interestingly, M only depends logarithmically on the number L of target properties and is independent of the actual system size. This logarithmic suppression can help to combat the curse of dimensionality. For instance, only roughly n/ϵ^2 samples suffice to ϵ-approximate all $N = 2^n$ outcome probabilities of the underlying state distribution.

Overall, stochastic quantum circuit simulation allows to avoid the increase of complexity from 2^n-vectors to $2^n \times 2^n$- matrices. However, the challenge remains to produce and process samples $|\tilde{\psi}_i\rangle$ (which still remain exponential in size). State-of-the-art quantum circuit simulators like [11]–[13] still severely suffer from the remaining exponential complexity.

IV. Proposed Solution

In this section, we present a solution that addresses the problems that still exists in stochastic quantum circuit simulation. To this end, we first briefly introduce the main ideas of our solution; followed by more detailed descriptions afterwards. Section V eventually shows that the concepts introduced here have a substantial impact on the performance and the scalability of stochastic quantum circuit simulation.

A. General Ideas

Stochastic quantum circuit simulation suffers from the fact that (1) the underlying concepts require exponentially large representations of vectors and matrices and (2) that, in order to determine accurate predictions (see Theorem 1), a sufficient number of simulation runs (with these exponential representations) need to be conducted—posing severe challenges with respect to memory and runtime. In this work, we are addressing these challenges with the following two key ideas:

- *Use Decision Diagrams for Individual Simulation Runs*: In the conventional realm, decision diagrams (such as proposed, e.g., in [31]–[33]) have found great application to tackle several (exponentially hard) problems. Even in the quantum realm, first approaches successfully exploiting them have been reported (see, e.g., [22], [24], [34]–[37]). We propose to use these promises to tackle the challenge of exponential complexity in individual simulation runs.
- *Exploit Concurrency Across Different Simulation Runs*: In stochastic quantum simulation, interesting properties are obtained by empirically averaging over a sufficient number of *independent* simulation runs. This setup facilitates concurrent implementation: use different cores for computing different simulation runs. Since accurate predictions are contingent on empirically averaging many samples (see Theorem 1), the potential of concurrent protocol execution on multi-core architectures is enormous.

These two ideas turn out to complement each other nicely. Concurrency is a default feature of the proposed high-level solution (Monte Carlo), while decision diagrams provide a tractable way for executing individual simulation runs. In the remainder of this section, details of these ideas are described and illustrated.

B. Using Decision Diagrams for Individual Simulation Runs

The general idea of decision diagram-based quantum circuit simulation is about uncovering and exploiting redundancies in the representation of states and operations. Doing so can result in potentially very compact representations, which in turn allows simulating quantum circuits that cannot be tackled using other simulation approaches anymore.

Representing a state vector as a decision diagram revolves around recursively splitting the vector into equally 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 $|0\rangle$, while the remaining 2^{n-1} entries would represent amplitudes where q_0 is $|1\rangle$. This is represented in a decision diagram by a node labeled q_0 with a left (right) successor which 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.

Example 4. *In Fig. 1a, the state vector $|\psi'\rangle$ from Example 2 is represented as both, vector and decision diagram*[1]*. The annotations of the vector representation indicate how it is decomposed for the decision diagram representation. In order to reconstruct an amplitude from the decision diagram, the edge weights of the corresponding path must be multiplied. For example, the amplitude of the state $|11\rangle$ (represented by the bold path in Fig. 1a) can be reconstructed by multiplying the edge weights of the root edge ($\frac{1}{\sqrt{2}}$) with the right edge of q_0 (1), as well as the right edge of q_1 (1), i.e., $\frac{1}{\sqrt{2}} \cdot 1 \cdot 1 = \frac{1}{\sqrt{2}}$.*

Matrix representations of quantum operations are represented as decision diagrams in a similar way. However, due to the square nature of matrices, they are split into four equally sized sub-parts. These parts are represented in a decision diagram by a node with four successor edges. The first one representing the upper left, the second the upper right, the third the lower left, and the fourth the lower right sub-matrix. The remaining decomposition steps are analogous to the case of a vector described above.

Example 5. *Fig. 1b provides a decision diagram representation for a Z-operation applied to the first qubit of a 2-qubit register. The annotations in the matrix representation indicate how the matrix is decomposed into the resulting decision diagram. The matrix entry highlighted bold in Fig. 1b can be reconstructed by multiplying the edge weights of the root edge 1 with the first edge of q_0 (-1), as well as the first edge of q_1 (1), i.e., $1 \cdot -1 \cdot 1 = -1$.*

Using these representations, operations such as matrix-vector multiplication can be executed so that, simulation of quantum circuits can be conducted. However, similar to the vector and matrix representation, the multiplication must also be decomposed with respect to the most significant qubit.

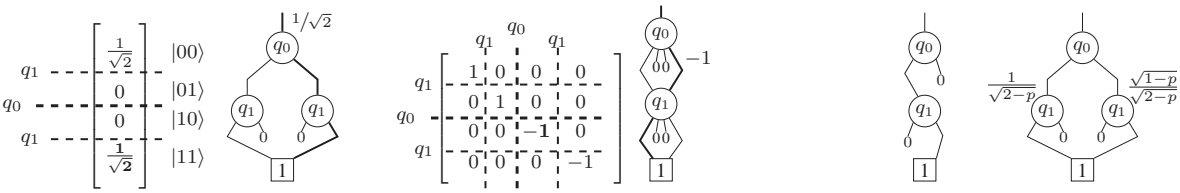
More precisely, consider a quantum register composed of n qubits given by $|\phi\rangle = q_0, q_1, \ldots, q_{n-1}$ (where q_0 represents the most significant qubit) and a unitary quantum operation U of size $2^n \times 2^n$. To multiply the operation U onto the state $|\phi\rangle$, they are split into two (in the case of the state vector) and four (in the case of the operation) equally sized parts. This leads to two sub-vectors of size 2^{n-1} and four sub-matrices of size $2^{n-1} \times 2^{n-1}$. This represents the modifications of U onto q_0 and is accordingly represented by a top node labeled q_0, with two successor nodes. Similar to the vector and matrix decomposition, this process is recursively repeated until vectors of size 2 and matrices of size 2×2 remain, which are multiplied. From the resulting new amplitudes, the new edge weights are calculated and equivalent sub-vectors are represented by the same node. Multiplications therefore mainly involves recursive traversals of the involved decision diagrams.

Finally, in order to consider error effects, we basically can re-use the concepts from above, i.e., we view error effects as operations, which are applied to the state with some probability. The outcome of such an erroneous operation is an *ensemble* of possible outcomes $\{(p_i, |\psi_i\rangle)\}$. Gate errors causing depolarization of qubits can be mimicked as illustrated in Example 3. Phase flip decoherence errors can be mimicked in a similar fashion, by applying a Z-operation to the qubit. The amplitude damping error cannot be simulated using simple gate operations. This is due to the fact that damping a qubit is not reversible, which is why it cannot be expressed using reversible (unitary) operations. This makes it so, that the probability of applying the error is influenced by the state the error is applied to, as illustrated in the following example.

Example 6. *Consider the state $|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$ from Example 2 (shown in Fig 1a) and suppose that it is subject to amplitude damping. Here, things get more interesting. If amplitude damping affects the first qubit with probability p, its action is given by the matrices $\mathrm{A}_0 = \begin{bmatrix} 0 & \sqrt{p} \\ 0 & 0 \end{bmatrix}$ and $\mathrm{A}_1 = \begin{bmatrix} 1 & 0 \\ 0 & \sqrt{1-p} \end{bmatrix}$ [2]. But, it is not the error probability p (alone) that matters. In contrast to depolarizing and phase flip errors, amplitude damping is manifestly state-dependent. In order to get the probability for applying either A_0 or A_1, they have to be applied to $|\psi'\rangle$. Applying A_0 to $|\psi'\rangle$ results in a state vector whose squared norm is $\frac{p}{2}$, which is also the probability that A_0 is applied. Analogously, the probability for applying A_1 is $1 - \frac{p}{2}$. Depending on those probabilities, one state is randomly chosen and normalized, while the other one is discarded. Thus damping the first qubit with probability p results in the ensemble $\{(\frac{p}{2}, |01\rangle), (1 - \frac{p}{2}, \frac{1}{\sqrt{2-p}}|00\rangle + \frac{\sqrt{1-p}}{\sqrt{2-p}}|11\rangle)\}$ (their decision diagram representations are also given in Fig. 1c).*

Using all that, the concepts for stochastic quantum circuit simulation as reviewed in Section III can be realized by means of decision diagrams. More precisely, recall from Section II that for simulating quantum circuits we need means to represent vectors and matrices for states and operations, respectively. Additionally, we need some means of applying operations to states (either for applying quantum operations or error effects). Having all that, the stochastic approach presented in Section III—which allows to apply error operations probabilistically—can be used in a straightforward fashion. Since decision diagrams often allow to represent all these entities and to conduct all these operations in a much more compact and efficient fashion, a big challenge of existing approaches for stochastic quantum circuit simulation is addressed.

[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—indicating that amplitudes of all possible states represented by this part of the decision diagram are zero.



(a) Vector representation (b) Matrix representation (c) Possible states after damping q_0

Fig. 1: Decision diagram representation of states

C. Exploiting Concurrency Across Different Simulation Runs

As detailed above, decision diagrams provide a powerful data structure that often helps to escape exponential memory requirements. This facilitates the faithful execution of moderate-scale quantum simulation. At the same time, however, decision diagrams can hardly exploit concurrency thus far [38]. This is in stark contrast to state-of-the art quantum simulators (such as [11]–[19], [25]) which heavily make use of concurrent executions (e.g., during matrix-vector multiplication).

Stochastic quantum simulation, however, is an interesting use case where the apparent trade-off between optimizing memory (through decision diagrams) and exploiting concurrency (to accelerate matrix-vector multiplication) can be resolved by different means: Simply use different cores to compute *independent* simulation runs (samples). Given that accurate predictions require a sufficient number of independent samples (see Theorem 1), the potential of exploiting concurrency across different simulation runs—rather than within individual runs—is enormous.

It is worthwhile to point out that concurrent execution is a general feature of Monte-Carlo-type approximations and well known. Stochastic quantum simulation is merely an interesting special case. Implementations of stochastic quantum simulations, e.g., in [11]–[13], do not seem to utilize this potential yet (most likely, because most existing approaches rely on exponentially large vector and matrix representations limiting the potential of having several runs of this size in parallel and, hence, exploiting concurrency during the matrix-vector multiplications seemed to be the more feasible approach). With the proposed approach, both (memory-efficient representations *and* concurrent executions) can be exploited.

V. Evaluation

In order to empirically evaluate the performance of the proposed stochastic error simulation approach, we implemented the concepts described above in C++ (using the open-source decision diagram package taken from [39]). Afterwards, we compared the resulting performance against other available state-of-the-art stochastic simulators, namely the *LinAlg* simulator from the *Atos Quantum Learning Machine* (QLM) [13] and the *statevector* simulator from IBM's *Qiskit* [12].

We considered different benchmark sets: First, we evaluated all simulation approaches using the *Entanglement* circuit (an algorithm generating the GHZ state), as well as the *Quantum Fourier Transform* (QFT, [2]), with an increasing number of qubits. By this, we considered typical use cases incorporating quantum-mechanical effects such as superposition and entanglement in a scalable fashion (i.e., with an increasing number of qubits). Second, we evaluated all simulation approaches using the circuits from the benchmark suite *QASMBench* (taken from [40]), which contains a broad range of different quantum algorithms.

For all benchmarks, we considered all errors discussed in Section II-B, i.e., gate errors, as well as decoherence errors. More precisely, we applied a depolarization error with 0.1 % probability, an amplitude damping (T1) error with 0.2 % probability, and a phase flip error (T2) with 0.1 % probability to the gate/qubit. The errors have been simulated using the stochastic approach presented in Section III with a total of $M = 30,000$ iterations for each benchmark (using Theorem 1, this corresponds to tracking 1000 properties with an error margin of < 0.01 and a confidence of 95 %).

Table I summarizes the results of our evaluation. More precisely, Table Ia, Table Ib, and Table Ic provide the results for the entanglement benchmark, the QFT benchmark, and the QASMBench benchmarks, respectively. In each table, we provide the number n of qubits, as well as the required runtime for each simulation approach in seconds. Note that, due to space limitations, only a selection of the QASMBench benchmarks is explicitly listed. Out of the 53 benchmarks, 4 have been omitted as they could not be simulated by either simulation approach within the time limit of 1 hour. 39 benchmarks have been omitted because their differences in runtime between the simulation approaches remained rather small. Furthermore, note that we do not list any results from Atos' QLM simulator for the QASMBench benchmarks, since those circuits are only provided in the OpenQASM format, which is not supported by the QLM simulator.

The results show the improved performance of the proposed simulation approach compared to the state-of-the-art approaches by Atos and IBM for certain algorithms. For the entanglement and QFT benchmarks, a substantially better scalability with respect to the number of qubits can be reported. For the QASMBench benchmarks, the proposed solution reaches its limit and gives longer runtimes for the *ising*, *yqe_uccsd*, and *cc* circuits. In the other cases, the circuits could be simulated faster—some algorithms even by several orders of magnitude.

VI. Conclusion

Quantum circuit simulation is an important research area. However, many available quantum circuit simulators simplify the problem by simulating *perfect* quantum computers. Due to the fragile nature of quantum systems, quantum computers are always subject to errors during their calculations. Simulators which allow the consideration of errors during the simulation often suffer from the exponential complexity of vectors and matrices required for the simulation. We addressed this issue by implementing a stochastic circuit simulator, which exploits the compact representations of vectors and matrices offered by decision diagrams and utilizes concurrent executions for an efficient generation of samples. Evaluations and comparisons against state-of-the-art simulators by IBM and Atos show the improved scalability and efficiency of the proposed solution for certain applications.

TABLE I: Evaluation results

(a) Entanglement circuits

n	Qiskit [s]	QLM [s]	Proposed [s]
21	1075.74	22.05	**0.58**
22	2247.9	42.57	**0.88**
23	>3600	155.22	**0.68**
:	>3600	:	:
27	>3600	1186.58	**0.78**
28	>3600	2251.19	**0.89**
29	>3600	>3600	**0.88**
:	:	:	:
63	>3600	>3600	**2.61**
64	>3600	>3600	**2.72**

(b) QFT circuits

n	Qiskit [s]	QLM [s]	Proposed [s]
12	20.61	1679.412	**2.41**
13	31.07	2338.356	**3.34**
14	60.13	>3600	**4.25**
:	:	:	:
17	834.95	>3600	**7.22**
18	2115.71	>3600	**8.97**
19	>3600	>3600	**10.17**
:	:	:	:
63	>3600	>3600	**377.68**
64	>3600	>3600	**402.65**

(c) QASMBench circuits

Name	n	Qiskit [s]	Proposed [s]
basis_trotter	4	112.69	**47.85**
vqe_uccsd	6	181.82	**88.59**
vqe_uccsd	8	**520.37**	3600.00
ising	10	**20.48**	282.23
seca	11	12.32	**1.29**
sat	11	37.33	**2.12**
multipler	15	136.83	**0.78**
bigadder	18	829.33	**0.98**
cc	18	**236.60**	3600.00
bv	19	448.14	**211.35**

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).

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

[3] I. Kassal, S. Jordan, P. Love, M. Mohseni, and A. Aspuru-Guzik, "Polynomial-time quantum algorithm for the simulation of chemical dynamics," *Proc. of the National Academy of Sciences*, vol. 105, no. 48, pp. 18 681–18 686, 2008.

[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] F. Arute *et al.*, "Quantum supremacy using a programmable superconducting processor," *Nature*, vol. 574, no. 7779, pp. 505–510, 2019.

[8] Martin Giles, "IBM's new 53-qubit quantum computer is the most powerful machine you can use," https://www.technologyreview.com/2019/09/18/132956/ibms-new-53-qubit-quantum-computer-is-the-most-powerful-machine-you-can-use/, 2019, Accessed: 2020-09-10.

[9] F. Lardinois, "IBM publishes its quantum roadmap, says it will have a 1,000-qubit machine in 2023," https://techcrunch.com/2020/09/15/ibm-publishes-its-quantum-roadmap-says-it-will-have-a-1000-qubit-machine-in-2023/, 2020, Accessed: 2020-09-15.

[10] 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

[11] "Forest SDK," https://pyquil-docs.rigetti.com/en/stable/index.html, 2020, accessed: 2020-07-22.

[12] H. Abraham *et al.*, "Qiskit: An open-source framework for quantum computing," 2019.

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

[14] N. Khammassi, I. Ashraf, X. Fu, C. Almudever, and K. Bertels, "QX: A high-performance quantum computer simulation platform," in *Design, Automation and Test in Europe*, 2017.

[15] D. Wecker and K. Svore, "LIQUi|>: A software design architecture and domain-specific language for quantum computing," *arXiv:1402.4467*, 2014.

[16] "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.

[17] T. Jones, A. Brown, I. Bush, and S. Benjamin, "Quest and high performance simulation of quantum computers," *arXiv preprint arXiv:1802.08032*, 2018.

[19] B. Villalonga, S. Boixo, B. Nelson *et al.*, "A flexible high-performance simulator for verifying and benchmarking quantum circuits implemented on real hardware," *npj Quantum Information*, vol. 5, no. 1, 2019.

[18] 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.

[20] 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.

[21] G. Vidal, "Efficient classical simulation of slightly entangled quantum computations," *Physical review letters*, vol. 91, no. 14, 2003.

[22] 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.

[23] 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.

[24] 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.

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

[26] J. Watrous, *The Theory of Quantum Information*. Cambridge University Press, 2018.

[27] S. Tannu and M. Qureshi, "Not All Qubits Are Created Equal," *arXiv:1805.10224*, 2018.

[28] Gambetta, Jay and Sheldon, Sarah, "Cramming more power into a quantum device," https://www.ibm.com/blogs/research/2019/03/power-quantum-device/, 2019, Accessed: 2020-09-10.

[29] M. Devoret and R. Schoelkopf, "Superconducting Circuits for Quantum Information: An Outlook," *Science*, vol. 339, pp. 1169–1174, 2013.

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

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

[32] S. Minato, "Zero-suppressed BDDs for set manipulation in combinational problems," in *Design Automation Conf.*, 1993, pp. 272–277.

[33] R. Drechsler, A. Sarabi, M. Theobald, B. Becker, and M. Perkowski, "Efficient representation and manipulation of switching functions based on ordered Kronecker functional decision diagrams," in *Design Automation Conf.*, 1994, pp. 415–419.

[34] G. F. Viamontes, M. Rajagopalan, I. L. Markov, and J. P. Hayes, "Gate-level simulation of quantum circuits," in *Asia and South Pacific Design Automation Conf.*, 2003, pp. 295–301.

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

[36] 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.

[37] 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.

[38] S. Hillmich, A. Zulehner, and R. Wille, "Concurrency in DD-based quantum circuit simulation," in *Asia and South Pacific Design Automation Conf.*, 2020.

[39] 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.

[40] 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