

Towards Hamiltonian Simulation with Decision Diagrams

Aaron Sander* Lukas Burgholzer† Robert Wille*§

*Chair for Design Automation, Technical University of Munich, Munich, Germany

†Institute for Integrated Circuits, Johannes Kepler University, Linz, Austria

§Software Competence Center Hagenberg (SCCH) GmbH, Hagenberg, Austria

aaron.sander@tum.de, lukas.burgholzer@jku.at, robert.wille@tum.de

<https://www.cda.cit.tum.de/research/quantum/>

Abstract—This paper proposes a novel approach to Hamiltonian simulation using *Decision Diagrams* (DDs), which are an exact representation based on exploiting redundancies in representations of quantum states and operations. While the simulation of Hamiltonians has been studied extensively, scaling these simulations to larger or more complex systems is often challenging and may require approximations or new simulation methods altogether. DDs offer such an alternative that has not yet been applied to Hamiltonian simulation. In this work, we investigate the behavior of DDs for this task. To this end, we review the basics of DDs such as their construction and present how the relevant operations for Hamiltonian simulation are implemented in this data structure—leading to the first DD-based Hamiltonian simulation approach. Based on several series of evaluations and comparisons, we then discuss insights about the performance of this complementary approach. Overall, these studies show that DDs indeed may offer a promising new data structure which, for certain examples, can provide orders of magnitudes of improvement compared to the state-of-the-art, yet also comes with its own, fundamentally different, limitations.

Index Terms—Decision Diagrams, Quantum Simulation, Tensor Networks, Quantum Computing

I. INTRODUCTION

Hamiltonian simulation is a powerful tool for understanding the behavior of complex physical systems [1]–[4] and related problems that can be mapped to Hamiltonians [5]–[7]. However, the development of efficient and accurate simulation methods remains a significant challenge due the exponential growth of the complex values needed to represent quantum states and operators. Hence, straightforward solutions such as statevector simulators [8]–[10] quickly run into scalability issues. Dedicated data structures such as tensor networks [11], [12] or neural network quantum states [13]–[15] remain limited in their ability to handle large and complex systems without reaching limits in memory or runtime requirements—motivating the research for further complementary methods which can overcome this limit for certain classes of problems [16], [17].

In this paper, we introduce *Decision Diagrams* (DDs) as a new data structure for simulating Hamiltonians. DDs have historically found great success in the domain of classical computing as an efficient means to represent and manipulate Boolean functions and, hence, classical circuits and systems [18]. Inspired by their application in classical computing, DDs were adapted to the quantum realm as a core data structure for quantum circuit simulation [19], [20], quantum circuit synthesis [21], [22], and quantum circuit verification [23]. They have, however, yet to be applied to more general

quantum systems. Their main advantage rests on exploiting redundancies in quantum states and operations which can lead to significant compression in the memory requirements needed to represent these structures and reduced the runtime necessary to perform calculations. In this paper, we shed light on their ability to simulate Hamiltonians and propose them as a complementary tool to other state-of-the-art methods.

To this end, we first review the basics on Hamiltonian simulation in Section II and describe the limits for simulating Hamiltonian systems using the current state-of-the-art in Section III. Motivated by that, we then introduce DDs as a novel, complementary approach in Section IV—aiming for a self-contained coverage without assuming any computer science background. Eventually, this leads to the first Hamiltonian simulation approach based on DDs.

Having that, we conducted several series of evaluations and comparisons to get insights about the performance of this complementary approach. The obtained results (summarized in Section V) clearly show that, for certain examples, DD-based Hamiltonian simulation can provide orders of magnitude of improvement compared to the state-of-the-art, i.e., methods based on sparse calculations, statevector simulators, and tensor networks. We also investigated the limitations of the proposed alternative, which, however, are fundamentally different for DDs compared to the current state-of-the-art methods—requiring a shift in perspective to fully utilize their benefits. We conclude from these investigations that DDs offer a promising new data structure for Hamiltonian simulation that is complementary to existing approaches and, hence, may continue to play an important role in the development of efficient and accurate simulation methods.

II. HAMILTONIAN SIMULATION

In this section, the main concepts of Hamiltonian simulation are reviewed. While the individual descriptions are kept brief, the interested reader is referred to [24]–[26] for more information.

A. Quantum Dynamics

Hamiltonian simulation is a computational technique used to simulate the dynamics of physical systems. The energy of a system is encoded into a mathematical object called a *Hamiltonian* that describes the interactions and states of a system. Based on this, the dynamics of a system can

be determined by solving the time-dependent Schrödinger equation for some quantum state $|\Psi\rangle$ and Hamiltonian H , i.e.,

$$\frac{d}{dt} |\Psi\rangle = -\frac{i}{\hbar} H |\Psi(0)\rangle. \quad (1)$$

This results in the unitary time evolution operator $U(t)$ that is used to calculate the state at some time t , i.e.,

$$|\Psi(t)\rangle = e^{-iHt} |\Psi(0)\rangle =: U(t) |\Psi(0)\rangle, \quad (2)$$

where $\hbar = 1$ for simplicity. It is then possible to calculate the dynamics of measurable physical quantities called *observables* using the expectation value, i.e.,

$$\langle O(t) \rangle = \langle \Psi(t) | O | \Psi(t) \rangle = \langle \Psi(0) | U^\dagger(t) O U(t) | \Psi(0) \rangle. \quad (3)$$

Examples of physical quantities that can be analyzed from observables include the energy of a system, magnetization, and the number of particles in a system [27].

B. Product Formula

The matrix exponential defining the time evolution operator in Eq. (2) is often difficult or impossible to compute, especially as it grows in size and complexity. In order to perform the time evolution within a controlled error bound, the *Lie product formula* can be used to break this operator into simpler, more efficient operations [24], [25]. The matrix exponential is decomposed into its constituent terms based on the *Baker-Campbell-Hausdorff* (BCH) formula, i.e.,

$$e^{xA} e^{xB} = e^{x(A+B) + \frac{1}{2}x^2[A,B] + \mathcal{O}(x^3)}, \quad (4)$$

where $[A, B]$ is the *commutator* of A and B defined by $[A, B] := AB - BA$.

This results in a matrix-analog to the exponential identity $e^a e^b = e^{(a+b)}$ which is commonly referred to as the first-order *Suzuki-Trotter decomposition*, i.e.,

$$e^{x(A+B)} = e^{xA} e^{xB} + \mathcal{O}(x^2). \quad (5)$$

From this, it directly follows that

$$e^{(A+B)} = \left(e^{\frac{A+B}{n}} \right)^n = \lim_{n \rightarrow \infty} \left(e^{\frac{A}{n}} e^{\frac{B}{n}} \right)^n. \quad (6)$$

Thus, the unitary time evolution for a Hamiltonian described by a sum of terms $H = A + B$ can be approximated by a sequence of discrete timesteps $\delta t = \frac{t}{n}$, i.e.,

$$U(t) = e^{-iHt} \approx \left(e^{-i\delta t A} e^{-i\delta t B} \right)^n = \left(U(\delta t) \right)^n, \quad (7)$$

where n is known as the *Trotter number*. According to Eq. (4), this approximation is exact whenever A and B commute since in that case $[A, B] = 0$. In general, the approximation error (known as *Trotter error*) scales with the number of non-commuting terms in the Hamiltonian H (according to Eq. (4)). Therefore, Hamiltonians with more non-commuting terms require more Trotter steps to compensate when using product formulas. However, this also means that product formulas perform especially well for simulating Hamiltonians with commuting or nearly-commuting terms and, in such cases, only require few Trotter steps.

For an overview on more precise error bounds that take into account more specific parameters such as system size, see [26].

C. Ising Model

This paper will primarily focus on the transverse-field *Ising model* as a representative of an important class of Hamiltonians. In its general form, it models the spins of particles and is described by the Hamiltonian

$$H = - \sum_{\langle i, j \rangle} J_{ij} \sigma_x^{[i]} \sigma_x^{[j]} - \sum_j g_j \sigma_z^{[j]}, \quad (8)$$

where J_{ij} is the interaction strength between nearest-neighbor spin pairs at sites $\langle i, j \rangle$ and g_j is an external field pointing perpendicular to the interactions at site j . It is a natural starting point for Hamiltonian simulation since it can be used to formulate many computationally hard problems, such as spin glasses, *Quadratic Unconstrained Binary Optimization problems* (QUBOs), or graph partitioning [5]. The following constrained version of this type of Hamiltonian will be used to illustrate all proposed concepts and methods throughout the remainder of this paper.

Example 1. The L -site finite 1D Ising chain is defined by

$$H = -J \sum_{\ell=0}^{L-2} \sigma_x^{[\ell]} \sigma_x^{[\ell+1]} - g \sum_{\ell=0}^{L-1} \sigma_z^{[\ell]}, \quad (9)$$

where the parameters are site-independent, i.e., $J_{ij} := J$ and $g_\ell := g$. Using the product formula, a single Trotter under this model has the form

$$U(\delta t) = \prod_{\ell=0}^{L-2} e^{iJ\delta t \sigma_x^{[\ell]} \sigma_x^{[\ell+1]}} \prod_{\ell=0}^{L-1} e^{iJ\delta t \sigma_z^{[\ell]}}, \quad (10)$$

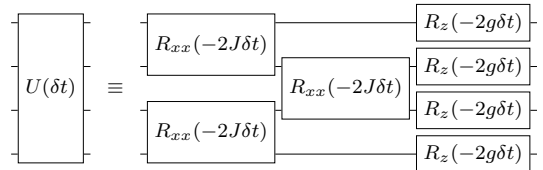
where each term is defined by the rotation gates

$$\begin{aligned} R_{xx}(\theta) &= e^{-i\frac{\theta}{2}(\sigma_x \otimes \sigma_x)} \\ &= \begin{pmatrix} \cos(\frac{\theta}{2}) & 0 & 0 & -i \sin(\frac{\theta}{2}) \\ 0 & \cos(\frac{\theta}{2}) & -i \sin(\frac{\theta}{2}) & 0 \\ 0 & -i \sin(\frac{\theta}{2}) & \cos(\frac{\theta}{2}) & 0 \\ -i \sin(\frac{\theta}{2}) & 0 & 0 & \cos(\frac{\theta}{2}) \end{pmatrix} \end{aligned} \quad (11)$$

and

$$\begin{aligned} R_z(\theta) &= e^{-i\frac{\theta}{2}\sigma_z} \\ &= \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix}. \end{aligned} \quad (12)$$

For a 4-site chain, this decomposition is equivalent to the circuit form with rotation gates applied first to even then odd sites; as sketched in the following figure:



D. Heisenberg Model

This paper also considers a related model called the Heisenberg model [28] which considers additional spin couplings

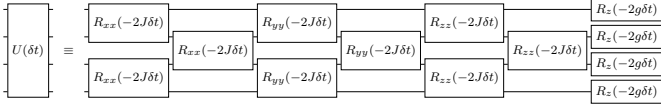
along the y- and z-axes. This is represented by the Hamiltonian, i.e.,

$$H = -J_x \sum_{\langle i,j \rangle} \sigma_x^{[i]} \sigma_x^{[j]} - J_y \sum_{\langle i,j \rangle} \sigma_y^{[i]} \sigma_y^{[j]} - J_z \sum_{\langle i,j \rangle} \sigma_z^{[i]} \sigma_z^{[j]} - h \sum_j \sigma_z^{[j]}. \quad (13)$$

Example 2. The XXX Heisenberg model is a subset of Heisenberg models such that $J_x = J_y = J_z$. The L -site finite 1D XXX Heisenberg chain is defined by

$$H = -J \left(\sum_{\ell=0}^{L-2} \sigma_x^{[\ell]} \sigma_x^{[\ell+1]} + \sum_{\ell=0}^{L-2} \sigma_y^{[\ell]} \sigma_y^{[\ell+1]} + \sum_{\ell=0}^{L-2} \sigma_z^{[\ell]} \sigma_z^{[\ell+1]} \right) - h \sum_{\ell=0}^{L-1} \sigma_z^{[\ell]}. \quad (14)$$

For a 4-site chain, a single Trotter step decomposition is equivalent to the circuit as shown in the following figure:



III. MOTIVATION

Hamiltonian simulation, reviewed above, is a crucial tool for understanding the behavior of quantum systems under different conditions. The fundamental physics research that uses quantum simulation can help us explore new phenomena and deepen our understanding of the fundamental workings of quantum systems. At the same time, quantum simulation can also have significant implications for material science and chemistry, enabling researchers to design and develop new materials and study chemical reactions at the molecular level.

By simulating the behavior of quantum systems using Hamiltonians, researchers can gain insights into the dynamics of quantum systems, and better understand how they will behave under different conditions. This is an important step in bridging theory and experimental results.

In addition, Hamiltonian simulation can help to accelerate the development of new quantum devices and quantum hardware. By simulating the behavior of these devices, researchers can optimize their designs and improve their performance.

Another important application of Hamiltonian simulation is in the optimization of complex systems. Many problems in fields such as machine learning [29] and logistics [30] can be mapped to Hamiltonians, allowing researchers to explore the energy landscapes and time-dynamics within given constraints. This can help to identify optimal solutions to complex problems and drive innovation in fields ranging from drug design to financial modeling.

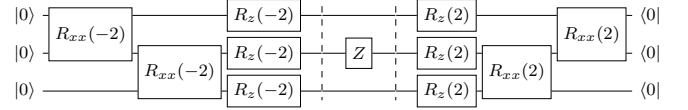
A. General Problem

However, despite this broad spectrum of applications, Hamiltonian simulation, at its core, relies on descriptions of quantum systems that grow exponentially with system size. Quantum states and operators, such as those that represent Hamiltonians, are often described by complex vectors and

matrices, respectively. For example, for a system consisting of L d -level systems, the quantum state $|\Psi\rangle$ can be represented by a vector in \mathbb{C}^{d^L} , and the operator O can be represented by a matrix in $\mathbb{C}^{d^L \times d^L}$. As the system size and complexity of interactions increase, these objects become increasingly computationally difficult to simulate due to the exponential growth in memory requirements and calculation runtime.

Therefore, there is a need for dedicated data structures that can efficiently simulate quantum systems on classical computers while scaling well with system size and complexity. This involves developing new algorithms and techniques that can represent the exponential growth in a more manageable way. This makes it possible to study more complex phenomena and design new materials and devices. As quantum technologies continue to advance, these classical methods will become increasingly important for simulating and understanding quantum systems in a wide range of applications.

Example 3. Consider again the scenario from Example 1 and, for simplicity, let $J = g = 1$. Assume that the system starts in the all-zero state, i.e., $|\Psi(0)\rangle = |0 \dots 0\rangle$ and that we are interested in the expectation value of the observable $O = \sigma_z^1$ at $t = 1$. Then, using a single Trotter step and, hence, $\delta t = \frac{t}{n} = 1$, this corresponds to the computations as shown in the following figure:



This can be applied for multiple values of t such that we can sample various points of the time evolution of the observable in time. Although simple in theory, calculating the expectation value for even a single time becomes computationally expensive for large system sizes.

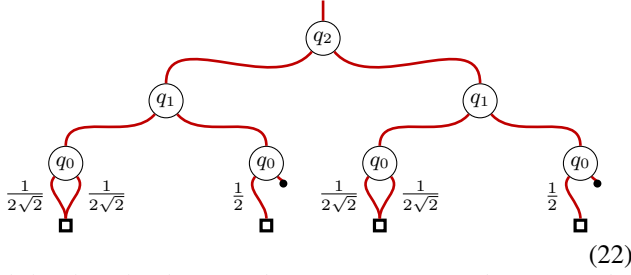
B. Related Work

In the past, various classical methods have been proposed to tackle the underlying complexity of this problem in different fashions. Each simulation method has its own strengths and limitations, and the choice of which method to use depends on the specific requirements of the simulation such that they can be seen as complementary to each other.

More precisely, a direct method that can save memory and computational time is by utilizing sparse vectors and matrices that only store non-zero terms [31]. This method does not require the decomposition of the time evolution operator into local operations, but it still suffers from exponential scaling according to system size. This method is easily accessible as many linear algebra packages such as SciPy have direct support for sparse data structures and operations such as the sparse matrix exponential [32].

Statevector simulation [8]–[10] is another method used for simulating quantum systems by decomposing large unitaries into circuits as done in the product formula described in Section II-B. This method is very useful for simulating small systems and is not limited by long-range interactions or deep circuits. They do, however, grow exponentially with system size as it is still necessary to store all the amplitudes of the statevector.

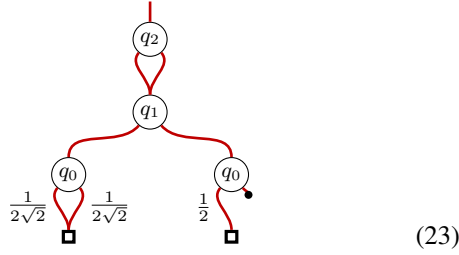
where $q_2, q_1, q_0 \in \{0, 1\}$. This directly translates to the decision diagram formalism:



Each level of the decision diagram consists of decision nodes with corresponding left and right successor edges. These successors represent the path that leads to an amplitude where the local quantum system (corresponding to the level of the node, annotated here with the labels) is in the $|0\rangle$ (left successor) or the $|1\rangle$ state (right successor).

At this point, this has just been a one-to-one translation between the statevector and a fancy graphical representation. The core, unique feature of decision diagrams is that their graph structure allows redundant parts to be merged in the representation instead of representing them repeatedly.

Example 6. Observe how, in the previous example, the left and the right successor of the top-level (q_0) node lead to exactly the same structure. As a result, the whole sub-diagram does not need to be represented twice, i.e.,

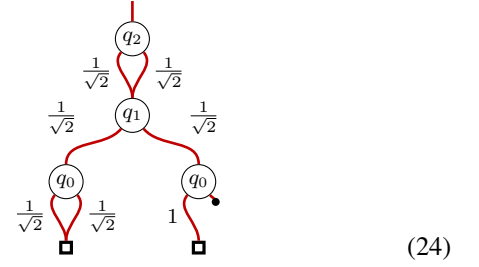


From a memory perspective, this reduction alone has compressed the overall memory required to represent the state by 50%.

Identifying redundancies in these kind of representations heavily depends on employing what is referred to as a *normalization scheme* for the decision diagrams nodes [41]. Such a normalization scheme makes sure that two decision diagram nodes that represent the same functionality do indeed have the same numerical structure. In computer science, this property is referred to as *canonicity* [41].

The most commonly used and practically relevant normalization scheme is to normalize the outgoing edges of a node by dividing both edge weights by the norm of the vector containing both edge weights and adjusting the incoming edges accordingly [42]. This normalizes the sum of the squared magnitudes of the outgoing edge weights to 1 and is consistent with the quantum semantics, where basis states $|0\rangle$ and $|1\rangle$ are observed after measurement with probabilities that are squared magnitudes of the respective weights. Normalization is recursively applied in a bottom-up fashion to ensure that every possible redundancy is being caught.

Example 7. Considering the decision diagram from the previous example, this results in the following normalized and reduced decision diagram:



The first two levels (q_2 and q_1) of the above diagram naturally encode that the respective sites have a 50/50 ($|1/\sqrt{2}|^2 = 0.5$) probability to be in $|0\rangle$ and $|1\rangle$. Meanwhile, the bottom level (q_0) encodes that the probability of q_0 depends on the state of q_1 . If q_1 is in the $|0\rangle$ state (following the left successor), then q_0 has probability 0.5 in both $|0\rangle$ or $|1\rangle$. If q_1 is in the $|1\rangle$ state (following the right successor), it is guaranteed that the remaining site is in the $|0\rangle$ state.

Overall, statevectors are represented as decision diagrams conceptionally equivalent to halving the vector in a recursive fashion until it is fully decomposed. The key idea is to exploit redundancies in the resulting diagrams to create a more compact representation. Some interesting properties that are worth pointing out:

- Decision diagrams can be initialized in their compact form (as, e.g., shown in Example 7). There is no need to create the maximally large decision diagram (as, e.g., shown in Example 5) at any point in a calculation.
- Determining a particular amplitude of the represented state corresponds to multiplying the edge weights along a single-path traversal from the top edge of the decision diagram (called its *root*) to a terminal node.
- The efficiency of decision diagrams is commonly measured by their *size*, i.e., the number of nodes in the decision diagram—the smaller the number of nodes, the higher the compaction achieved by the data structure. Note that the terminal (node) is typically not counted towards the size of a decision diagram.
- Any product state naturally has a decision diagram consisting of a single node per site. A compact DD does not, however, correlate to the state being trivial. Even highly-entangled states such as the GHZ state or the W state have decision diagrams whose size (i.e., the number of nodes) is linear in the number of sites.
- DDs are no "silver bullet." The worst case size of decision diagrams, corresponding to states with no redundancy, is still exponential in the number of sites. More specifically, a maximally large decision diagram has $1 + 2^1 + 2^2 + \dots + 2^{n-1} = 2^n - 1$ nodes.
- In implementation, redundancy in the complex edge weights is equivalent to comparison of floating point numbers within some tolerance.
- To reduce visual clutter in illustrations of decision diagrams, edge weights are commonly not annotated explicitly but their magnitude and phase is reflected in the

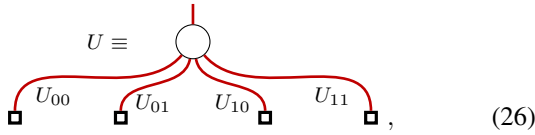
thickness and the color of the respective edge. In addition, to make the correspondence of the individual levels in a decision diagram to a system's sites more explicit, the nodes are frequently annotated with the site's index as an identifier. See [43] for further details on common techniques for the visualization of decision diagrams.

B. Representing Quantum Operators

Quantum operators are fundamentally described by (complex-valued) matrices. Just like moving from *Matrix Product State* (MPS) representations to *Matrix Product Operator* (MPO) representations in the domain of tensor networks, matrix decision diagrams are a natural extension to vector decision diagrams by an additional dimension. To this end, consider the base case of a 2×2 matrix U , i.e.,

$$U = \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} = U_{00} |0\rangle\langle 0| + U_{01} |1\rangle\langle 0| + U_{10} |0\rangle\langle 1| + U_{11} |1\rangle\langle 1|. \quad (25)$$

Then, the decision diagram representing this matrix has the structure



which again resembles the general structure of the matrix. Note that U_{ij} can be interpreted as the transformation of $|j\rangle$ to $|i\rangle$.

Example 8. The following shows decision diagram representations for selected single-qubit operations:

$$I = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \equiv \begin{array}{c} \text{circle} \\ \swarrow \quad \searrow \\ \text{square} \quad \text{square} \end{array} \quad X = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \equiv \begin{array}{c} \text{circle} \\ \swarrow \quad \searrow \\ \text{square} \quad \text{square} \end{array}$$

$$R_z(\theta) = \begin{pmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{pmatrix} \equiv \begin{array}{c} \text{circle} \\ \swarrow \quad \searrow \\ \text{square} \quad \text{square} \end{array} \equiv \begin{array}{c} \text{circle} \\ \swarrow \quad \searrow \\ \text{square} \quad \text{square} \end{array} \quad (27)$$

The last equivalence demonstrates how a common factor between the edge weights can be pulled out and attached to the incoming (root) edge.

The generalization to larger matrices works analogously to the vector case. To construct the decision diagram representing a matrix, the matrix is recursively split into quarters and the four elements correspond to the four successors of the node for representing that split. As for vector decision diagrams, a normalization scheme is applied to ensure that the resulting data structure is canonical and redundancy can be exploited. The conventional approach is to normalize all edge weights by the weight with the highest magnitude, selecting the leftmost one if multiple weights have the same magnitude. It is important to note that this ensures that all complex numbers within the decision diagram have a magnitude of at most 1, although this is subject to the implementation.

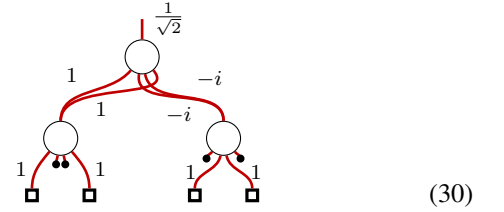
Example 9. Consider the maximally-entangling two-qubit R_{xx} rotation represented by the matrix

$$R_{xx}\left(\theta = \frac{\pi}{2}\right) = \frac{1}{\sqrt{2}} \begin{pmatrix} 1 & 0 & 0 & -i \\ 0 & 1 & -i & 0 \\ 0 & -i & 1 & 0 \\ -i & 0 & 0 & 1 \end{pmatrix}. \quad (28)$$

This matrix is equivalent to blocks of 2×2 matrices corresponding to the identity I and the Pauli- X matrix, i.e.,

$$R_{xx}\left(\theta = \frac{\pi}{2}\right) = \frac{1}{\sqrt{2}} \begin{pmatrix} I & -iX \\ -iX & I \end{pmatrix}. \quad (29)$$

The corresponding (already reduced) decision diagram has the following structure:



Notice how the decision diagram naturally resembles the structure of the matrix. The nodes at the bottom represent the identity and the X matrix (cf. Eq. (27)) while the node at the top encodes the redundancy of the upper-left and the bottom-right quadrant as well as the upper-right and the lower-left quadrant in Eq. (29). Similar to Example 6, exploiting redundancy has halved the overall memory requirement.

Again, some interesting properties to point out:

- Just as in the vector case, it is always possible to work with the reduced form of matrix decision diagrams right away, i.e., without ever constructing the exponentially-sized, maximally-large diagram.
- A maximally-large matrix decision diagram for an L -site two-level system has $\sum_{i=1}^L 4^{i-1} = \frac{1}{3}(4^L - 1)$ nodes.
- Decision diagrams are not limited to local interactions. Even long-range interactions between arbitrary sites typically emit compact representations as decision diagrams. For example, any two-site interaction between arbitrary sites can be represented as a decision diagram with at most $1 + 4(L - 1)$ nodes—an exponential reduction.
- Decision diagrams are not limited to two-qubit interactions either. For example, controlled quantum gates with arbitrarily many controls (such as the multi-controlled Toffoli gate) give rise to decision diagrams with a linear number of nodes.

C. Fundamental Operations on Decision Diagrams

Merely defining means for compactly representing any kind of state or operator does not yet allow one to perform Hamiltonian simulation. It is crucial to also define efficient means to work with or manipulate the resulting representations. In the following, we demonstrate how the most fundamental operations for Hamiltonian simulation can be carried out within the decision diagram formalism and how they scale. We will mostly focus on how operations are realized on

vectors, since the concepts extend from vectors to matrices in a straight-forward fashion.

The main concept throughout all of these schemes is to recursively break the respective operations down into sub-computations. This decomposition then naturally translates to the recursive decomposition of decision diagrams. As such, operations generally scale with the number of nodes in the involved decision diagrams.

Kronecker Product

The Kronecker product is necessary for creating product states as well as chaining together local operations. For vectors, it can be expressed as

$$|\Psi\rangle \otimes |\Phi\rangle = \begin{pmatrix} \Psi_0 \\ \Psi_1 \end{pmatrix} \begin{pmatrix} \Phi_0 \\ \Phi_1 \end{pmatrix} = \begin{pmatrix} \Psi_0 \Phi_0 \\ \Psi_0 \Phi_1 \\ \Psi_1 \Phi_0 \\ \Psi_1 \Phi_1 \end{pmatrix}. \quad (31)$$

In the decision diagram formalism, this is one of the simplest operations to perform and is done by simply replacing the terminal nodes of the first decision diagram with the root node of the second decision diagram. In case of the above example, this has the following form:

$$\begin{array}{c} \Psi \\ \swarrow \quad \searrow \\ \Psi_0 \quad \Psi_1 \end{array} \otimes \begin{array}{c} \Phi \\ \swarrow \quad \searrow \\ \Phi_0 \quad \Phi_1 \end{array} = \begin{array}{c} \Psi \\ \swarrow \quad \searrow \\ \Psi_0 \quad \Psi_1 \\ \Phi \\ \swarrow \quad \searrow \\ \Phi_0 \quad \Phi_1 \end{array} \quad (32)$$

As such, its complexity is linear in the number of nodes of the first decision diagram.

Addition

Standard vector addition can be recursively broken down according to

$$\begin{aligned} |\Psi\rangle + |\Phi\rangle &= \begin{pmatrix} \Psi_0 \\ \Psi_1 \end{pmatrix} + \begin{pmatrix} \Phi_0 \\ \Phi_1 \end{pmatrix} \\ &= w \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} + w' \begin{pmatrix} \alpha'_0 \\ \alpha'_1 \end{pmatrix} \\ &= \begin{pmatrix} w\alpha_0 + w'\alpha'_0 \\ w\alpha_1 + w'\alpha'_1 \end{pmatrix}, \end{aligned} \quad (33)$$

where w and w' are common factors of the terms in $|\Psi\rangle$ and $|\Phi\rangle$, respectively.

In the decision diagram formalism, this corresponds to a simultaneous traversal of both decision diagrams from their roots to the terminal (multiplying edge weights along the way until the individual amplitudes are reached) and back again

(accumulating the results of the recursive computations). More precisely,

$$\begin{array}{c} w \\ \swarrow \quad \searrow \\ \Psi_0 \quad \Psi_1 \end{array} + \begin{array}{c} w' \\ \swarrow \quad \searrow \\ \Phi_0 \quad \Phi_1 \end{array} = \begin{array}{c} ww' \\ \swarrow \quad \searrow \\ \Psi_0 \quad \Psi_1 \\ \Phi_0 \quad \Phi_1 \end{array} \quad (34)$$

where the dashed nodes represent the respective successor decision diagrams. Overall, this results in a complexity that is linear in the size of the larger decision diagram.

Matrix-Vector Multiplication

Matrix-vector multiplication can be handled in a very similar fashion as addition. Standard matrix-vector multiplication can be expressed as

$$\begin{aligned} U|\Psi\rangle &= \begin{pmatrix} U_{00} & U_{01} \\ U_{10} & U_{11} \end{pmatrix} \begin{pmatrix} \Psi_0 \\ \Psi_1 \end{pmatrix} \\ &= w \begin{pmatrix} u_{00} & u_{01} \\ u_{10} & u_{11} \end{pmatrix} w' \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \\ &= ww' \begin{pmatrix} u_{00} \cdot \alpha_0 + u_{01} \cdot \alpha_1 \\ u_{10} \cdot \alpha_0 + u_{11} \cdot \alpha_1 \end{pmatrix}. \end{aligned} \quad (35)$$

This implies that a multiplication boils down to four smaller multiplications and two additions. In the decision diagram formalism, this has the form

$$\begin{array}{c} w \\ \swarrow \quad \searrow \\ U_{00} \quad U_{01} \\ U_{10} \quad U_{11} \end{array} \cdot \begin{array}{c} w' \\ \swarrow \quad \searrow \\ \Psi_0 \quad \Psi_1 \end{array} = \begin{array}{c} ww' \\ \swarrow \quad \searrow \\ U_{00} \quad U_{01} \\ U_{10} \quad U_{11} \\ \Psi_0 \quad \Psi_1 \end{array} \quad (36)$$

where the dashed nodes again represent the respective successor decision diagrams. Overall, this results in a complexity that scales with the product of the size of both decision diagrams.

Inner Product

Computing the inner product of two vectors can be recursively broken down according to

$$\begin{aligned} \langle \Psi | \Phi \rangle &= (\Psi_0^* \quad \Psi_1^*) \begin{pmatrix} \Phi_0 \\ \Phi_1 \end{pmatrix} \\ &= w^* (\alpha_0^* \quad \alpha_1^*) w \begin{pmatrix} \alpha_0 \\ \alpha_1 \end{pmatrix} \\ &= w^* w (\alpha_0^* \alpha_0 + \alpha_1^* \alpha_1) \end{aligned} \quad (37)$$

This implies that the inner product boils down to two smaller inner product computations and adding the results. As with the matrix-vector multiplication, this is done recursively for each level of the decision diagram. In the decision diagram formalism, this has the following form

$$\begin{aligned}
 & \left\langle \begin{array}{c} \alpha_0 \\ \alpha_1 \end{array} \middle| \begin{array}{c} w \\ \Psi_0 \\ \Psi_1 \end{array} \right\rangle + \left\langle \begin{array}{c} \alpha'_0 \\ \alpha'_1 \end{array} \middle| \begin{array}{c} w' \\ \Phi_0 \\ \Phi_1 \end{array} \right\rangle \\
 &= w^* w' \left(\left\langle \begin{array}{c} \alpha_0 \\ \alpha_1 \end{array} \middle| \begin{array}{c} \alpha'_0 \\ \alpha'_1 \end{array} \right\rangle + \left\langle \begin{array}{c} \alpha_0 \\ \alpha_1 \end{array} \middle| \begin{array}{c} \alpha'_0 \\ \alpha'_1 \end{array} \right\rangle \right), \quad (38)
 \end{aligned}$$

Overall, this results in a complexity that, just as addition, scales linearly with the size of the larger decision diagram.

Expectation Value

Computing the expectation value of some observable O for a given state $|\Psi\rangle$ can be reduced to a matrix-vector multiplication and an inner product computation as follows:

$$\langle \Psi | O | \Psi \rangle = \langle \Psi | \left(O | \Psi \rangle \right) = \langle \Psi | \tilde{\Psi} \rangle \quad (39)$$

This directly translates to decision diagrams via Eq. (36) and Eq. (38). The resulting computation has an overall complexity that scales with the product of the size of both decision diagrams.

V. EXPERIMENTAL EVALUATIONS

Using the concepts presented above, we implemented the first Hamiltonian simulation approach based on decision diagrams. To this end, we used the open-source DD package available at https://github.com/cda-tum/dd_package (which is part of the *Munich Quantum Toolkit*, MQT) and extended it to support all necessary gates (such as the various two-qubit rotation gates) and operations (such as the expectation value). Afterwards, we conducted several series of evaluations and comparisons to get insights about the performance of the resulting DD-based Hamiltonian simulation. More precisely, we first analyzed different scaling characteristics of decision diagrams themselves followed by comparing their performance to other state-of-the-art techniques and considering selected best case scenarios. In this section, we summarize the obtained results and findings.

A. Application of DDs to Hamiltonian Simulation Circuits

In a first series of evaluations, we first analyzed the behavior of DDs in simulating various combinations of rotation angles in the Hamiltonian simulation circuits for the Ising and Heisenberg models described in Section II-A. The results are shown in Fig. 1. More precisely, we generated heatmaps, which we call *redundancy landscapes*, that show the number of nodes in the DD after applying each circuit. This was performed for $n = 1$ and $n = 2$ Trotter steps to a system size $L = 12$ initialized in the $|0 \dots 0\rangle$ state. The x- and y-axis correspond to the angle of the single- and two-site rotations, respectively.

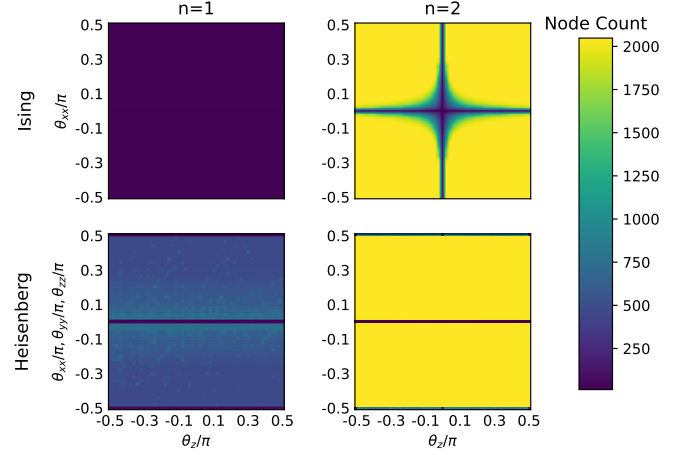


Fig. 1. Redundancy landscapes for the Ising and Heisenberg model with $L = 12$ for $n = 1$ and $n = 2$ Trotter step applications of the selected angles.

The node count scaling is normalized across both graphs—from maximal compaction (dark blue; linear regime) to almost no compaction (yellow; exponential regime).

These landscapes provide an abstract understanding of how each configuration of angles affects the size of the decision diagram. By using generalized angles based on the parameters of the Ising and Heisenberg Hamiltonians, these landscapes can also help identify the model parameters and timestep sizes that lead to compact decision diagrams.

From the results, it can be seen that the Ising model stays compact—for one Trotter step, regardless of the angle combinations—achieving almost maximal compaction. For two Trotter steps, the landscape begins to saturate such that there is almost no compaction for large angles. However, for small angles, the size remains moderate even for multiple Trotter steps.

For the Heisenberg model, a single Trotter step causes larger, but still not maximally large, decision diagrams compared to the Ising model. However, multiple Trotter steps causes the node count of the decision diagram to quickly grow with respect to the magnitude of the angles.

This suggests that DDs are likely to remain compact for Ising models and their derivatives, but require more work for simulating the Heisenberg model.

B. Node Count based on System Size and Trotter Number

In a second series of evaluations, we evaluated the node count in the DD as various number of Trotter steps are applied to several system sizes. The results according to the Ising and Heisenberg models are depicted in Fig. 2. In this figure, multiple Trotter steps are applied to systems ranging from size $L = 2$ to $L = 10$. The circuits' angles are fixed and were selected based on the redundancy landscapes seen in the previous section. The Ising model shown has a normalized interaction parameter of $J = 1$ and weak transverse field of $g = 0.001$. The Heisenberg model has interaction parameters of $J_x = J_y = J_z = 1$ and a field of $h = 1$. The timestep size in both cases is $\delta t = 0.1$.

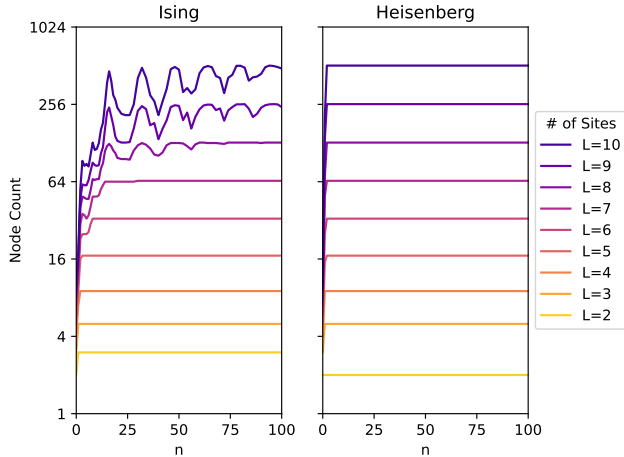


Fig. 2. Growth of node count in the decision diagrams for the Ising Model ($J = 1, g = 0.001$) and Heisenberg model ($J_x = J_y = J_z = 1, h = 1$) with timestep size $\delta t = 0.1$.

Several insights can be drawn from the plotted results. The initial slope for the first few Trotter steps has the greatest impact on scaling, with a maximum node count of 2^{L-1} in both models. This represents a 50% compaction, but still scales exponentially with system size.

The Ising model exhibits an oscillation in node count, corresponding to times when the state is more redundant than others. This oscillation becomes larger as the system size grows, indicating that larger systems have more redundancies and more opportunities for significant compression. The lower and upper bounds of this oscillation converge as the number of Trotter steps increases.

In contrast, the Heisenberg model converges to its maximum node count after only two Trotter steps—aligning with the growth shown in the redundancy landscapes. At this timescale, the selected Heisenberg model parameters do not exhibit many redundancies in its DD representation.

These results suggest, again, that the Ising model can benefit from significant compression over long timescales—particularly for large systems. The oscillation in its node count also supports the need to engineer redundancy to maximize the benefits of the DDs. On the other hand, the Heisenberg model does not exhibit much redundancy.

C. Comparison with Related Work

In a third series of evaluations, we compare the runtime scaling of the proposed DD-based Hamiltonian simulation against other simulation methods. More specifically, we compare against the time needed for an exact calculation with sparse methods implemented in SciPy [32], the Qiskit statevector simulator [8], and tenpy TEBD implementation [44]. These results are shown in Fig. 3. Each plot shows the time for each method to perform the time evolution and calculation of the expectation value of a local observable $\langle \sigma_z \rangle$ at the center of the system. These results are averaged over 10 runs. This is done with the Ising model with parameters $J = 1, g = 0.001$, and timestep size $\delta t = 0.1$, but similar results are seen for

other combinations such that this can be seen as a general guideline.

The results show that, for a single Trotter step, decision diagrams outperform both the Qiskit statevector simulator and tenpy TEBD, regardless of the system size. As the number of Trotter steps increases, however, the advantage of decision diagrams diminishes for larger systems. We notice that the runtime begins to converge with that of the sparse method. These findings suggest that decision diagrams are most advantageous for problems that can be solved in a single Trotter step. For multiple Trotter steps, they still provide a computational advantage for small systems, but the runtime converges, in the worst case, to the time required for a sparse method to compute the exact result.

These results show that DDs can indeed serve as a complementary alternative to current Hamiltonian simulation methods. As these results are based on the current state-of-the-art implementation of decision diagrams, it is expected that future work will lead to improved scaling for many Trotter steps and different models.

D. Selected Examples

In a final series of evaluations, we considered what we observed to be current best case scenarios for the considered DD-based Hamiltonian simulation, i.e., a 5-site Ising model and a 1000-site nearest-neighbor (Edwards-Anderson) spin glass chain without a field. These cases correspond to the results seen in Section V-C, i.e., small systems regardless of the number of Trotter steps and problems that can be solved in a single Trotter step, respectively.

The results of the 5-site Ising model can be seen in Fig. 4a. We show the time evolution of the two-site correlation function between the two ends of a 5-site chain $\langle \sigma_x^{[0]} \sigma_x^{[4]} \rangle$, evolving under an Ising model with parameters $J = 1, g = 0.001$. This plot contains 100 sampled points, each Trotterized with a timestep $\delta t = 0.1$, i.e., $Jt = 0.1$ requires 1 Trotter step and $Jt = 10$ requires 100 Trotter steps. The runtime plot clearly shows that DDs outperform the methods typically used for a system of this size by an order of magnitude.

Fig. 4b shows the results of the spin glass model. This model has Ising interactions along one direction without a field. This is equivalent to the generalized Ising model presented in Eq. (8) such that the parameters J are chosen randomly from a Gaussian distribution with mean 0 and standard deviation 1. This means the terms of the Hamiltonian commute which eliminates the Trotter error in Eq. (4). The time evolution of a local observable $\langle \sigma_z^{[499]} \rangle$ at the center of the chain is plotted with 100 sampled points for timestep $\delta t = 0.1$. Each point is calculated with a single Trotter step.

For systems of this size, tensor networks are currently the state-of-the-art method, as sparse methods and statevector simulators grow too large to be used efficiently. In this example, we also see that DDs outperform tensor networks by an order of magnitude.

These results indicate that the current implementation of DDs offers a significant computational advantage over other methods for small systems as well as for problems that can be solved in one Trotter step.

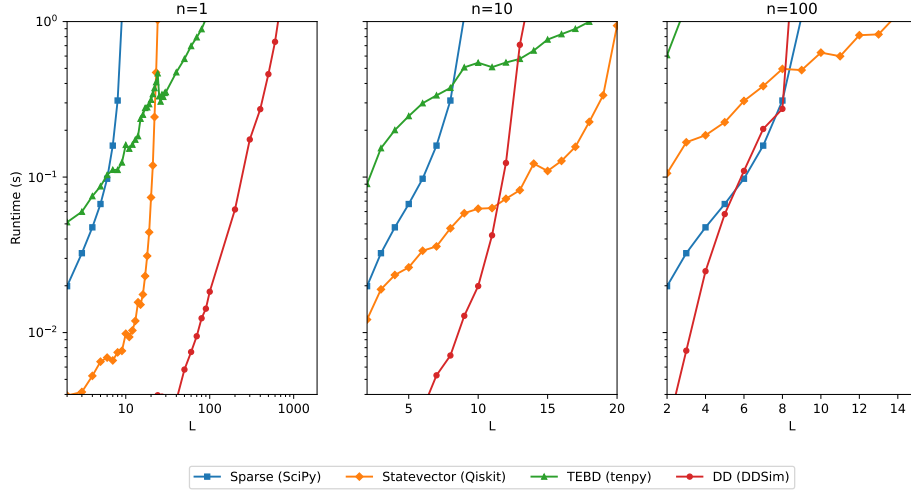
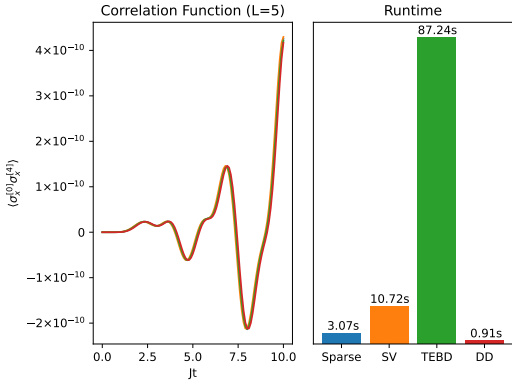
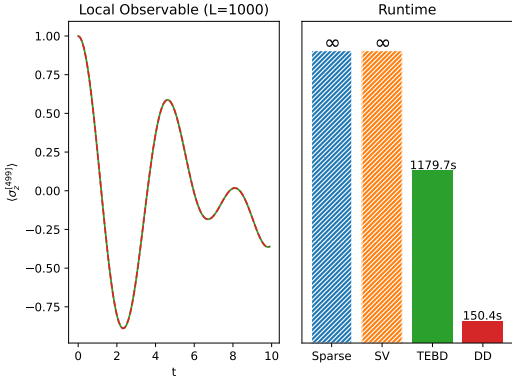


Fig. 3. Runtime scaling of various methods for simulating the Ising model $J = 1$, $g = 0.001$, and timestep size $\delta t = 0.1$. Each corresponds to this being performed with n Trotter steps.



(a) Time evolution of a two-site correlation function between the ends of a chain under the Ising model $J = 1$, $g = 0.001$ for $L = 5$ sites



(b) Time evolution of an Edwards-Anderson spin glass model with Gaussian distributed interaction parameters J with mean 0 and standard deviation 1

Fig. 4. Selected examples and their runtime requirement for different methods

VI. CONCLUSION

In this work, we proposed *Decision Diagrams* (DDs) as a promising new data structure for Hamiltonian simulation. The obtained results show that DDs can efficiently handle highly

redundant models, such as Ising-type models, surpassing statevector simulators and tensor networks in memory and runtime requirements. For problems solvable in a single Trotter step, DDs can efficiently scale to large systems, while for multiple Trotter steps, DDs show an advantage over other methods for small systems. As the number of Trotter steps increases, DDs eventually converge to the runtime requirements of sparse methods. However, our analysis of more complex models, such as the Heisenberg model, suggests that these models do not show as promising results, likely due to the lower level of redundancy. Further research is needed to explore DDs' potential for more complex models.

Still, these initial results already suggest that DDs could hold promise for problems that can be formulated as Ising models, such as QUBOs and graph problems. Additionally, preliminary application of DDs to long-range and higher-dimensional models have shown promise, although they exhibit redundancies that are not captured in the current decision diagram formalism. Therefore, future research could explore these higher-dimensional models and find ways to engineer redundancy in problem formulations.

We also acknowledge that the limitations of DDs are fundamentally different from those of current state-of-the-art methods, requiring a shift in perspective to fully utilize their benefits. Using DDs for Hamiltonian simulation opens up the possibility of implementing techniques from the field of graph theory, potentially allowing for sophisticated approximation techniques that could improve the scaling. The creation of approximation methods and finding ways to engineer redundancy in problem formulations are a promising next step for DDs and are left to future research.

In conclusion, this work demonstrates that DDs offer a promising new data structure for Hamiltonian simulation, especially for problems formulated with redundancies in mind. Future research can focus on refining the implementation of DDs and exploring their application to more complex models and problems. We anticipate that DDs will continue to play an important role in the development of efficient and accurate simulation methods.

REFERENCES

- [1] J. Ignacio Cirac and Peter Zoller, “Goals and opportunities in quantum simulation,” en, *Nature Physics*, vol. 8, no. 4, pp. 264–266, 2012, Number: 4 Publisher: Nature Publishing Group. DOI: 10.1038/nphys2275.
- [2] Richard P. Feynman, “Simulating physics with computers,” en, *International Journal of Theoretical Physics*, vol. 21, no. 6, pp. 467–488, 1982. DOI: 10.1007/BF02650179.
- [3] I. M. Georgescu, S. Ashhab, and Franco Nori, “Quantum simulation,” *Reviews of Modern Physics*, vol. 86, no. 1, pp. 153–185, 2014, Publisher: American Physical Society. DOI: 10.1103/RevModPhys.86.153.
- [4] Andrew J. Daley *et al.*, “Practical quantum advantage in quantum simulation,” en, *Nature*, vol. 607, no. 7920, pp. 667–676, 2022, Number: 7920 Publisher: Nature Publishing Group. DOI: 10.1038/s41586-022-04940-6.
- [5] Andrew Lucas, “Ising formulations of many NP problems,” *Frontiers in Physics*, vol. 2, 2014. DOI: 10.3389/fphy.2014.00005.
- [6] Zhidong Zhang, “Mapping between Spin-Glass Three-Dimensional (3D) Ising Model and Boolean Satisfiability Problem,” en, *Mathematics*, vol. 11, no. 1, p. 237, 2023, Number: 1 Publisher: Multidisciplinary Digital Publishing Institute. DOI: 10.3390/math11010237.
- [7] Hannes Leipold and Federico M. Spedalieri, “Constructing driver Hamiltonians for optimization problems with linear constraints,” en, *Quantum Science and Technology*, vol. 7, no. 1, p. 015013, 2021, Publisher: IOP Publishing. DOI: 10.1088/2058-9565/ac16b8.
- [8] Qiskit contributors, *Qiskit: An open-source framework for quantum computing*, 2023. DOI: 10.5281/zenodo.2573505.
- [9] Ville Bergholm *et al.* (2022). “PennyLane: Automatic differentiation of hybrid quantum-classical computations.” arXiv: arXiv:1811.04968, preprint.
- [10] *Cirq: A python framework for creating, editing, and invoking Noisy Intermediate Scale Quantum (NISQ) circuits*. [Online]. Available: <https://github.com/quantumlib/Cirq>.
- [11] Román Orús, “Tensor networks for complex quantum systems,” en, *Nature Reviews Physics*, vol. 1, no. 9, pp. 538–550, 2019, Number: 9 Publisher: Nature Publishing Group. DOI: 10.1038/s42254-019-0086-7.
- [12] Jacob C. Bridgeman and Christopher T. Chubb, “Hand-waving and Interpretive Dance: An Introductory Course on Tensor Networks,” en, *Journal of Physics A: Mathematical and Theoretical*, vol. 50, no. 22, p. 223001, 2017, arXiv:1603.03039 [cond-mat, physics:hep-th, physics:quant-ph]. DOI: 10.1088/1751-8121/aa6dc3.
- [13] Giuseppe Carleo and Matthias Troyer, “Solving the quantum many-body problem with artificial neural networks,” *Science*, vol. 355, no. 6325, pp. 602–606, 2017. DOI: 10.1126/science.aag2302.
- [14] Markus Schmitt and Markus Heyl, “Quantum many-body dynamics in two dimensions with artificial neural networks,” *Physical Review Letters*, vol. 125, no. 10, p. 100503, 2020. DOI: 10.1103/PhysRevLett.125.100503.
- [15] Irene López Gutiérrez and Christian B. Mendl, “Real time evolution with neural-network quantum states,” *Quantum*, vol. 6, p. 627, 2022. DOI: 10.22331/q-2022-01-20-627.
- [16] Xiaosi Xu *et al.*, *A Herculean task: Classical simulation of quantum computers*, arXiv:2302.08880 [quant-ph], 2023. DOI: 10.48550/arXiv.2302.08880.
- [17] Yiqing Zhou, E. Miles Stoudenmire, and Xavier Waintal, “What Limits the Simulation of Quantum Computers?” *Physical Review X*, vol. 10, no. 4, p. 041038, 2020, Publisher: American Physical Society. DOI: 10.1103/PhysRevX.10.041038.
- [18] S. Minato, “Zero-suppressed BDDs for set manipulation in combinatorial problems,” in *Design Automation Conf.*, 1993, pp. 272–277.
- [19] Alwin Zulehner and Robert Wille, “Advanced simulation of quantum computations,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2019.
- [20] Robert Wille, Stefan Hillmich, and Lukas Burgholzer, “Decision Diagrams for Quantum Computing,” in *Design Automation of Quantum Computers*, 2023.
- [21] Mehdi Saeedi, Robert Wille, and Rolf Drechsler, “Synthesis of quantum circuits for linear nearest neighbor architectures,” *Quantum Information Processing*, vol. 10, no. 3, pp. 355–377, 2011. DOI: 10.1007/s1128-010-0201-2. arXiv: 1110.6412.
- [22] Philipp Niemann, Robert Wille, and Rolf Drechsler, “Efficient synthesis of quantum circuits implementing Clifford group operations,” in *Asia and South Pacific Design Automation Conf.*, 2014, pp. 483–488.
- [23] Robert Wille and Lukas Burgholzer, “Verification of Quantum Circuits,” in *Handbook of Computer Architecture*, Springer Nature Singapore, 2022, pp. 1–28. DOI: 10.1007/978-981-15-6401-7_43-1.
- [24] Seth Lloyd, “Universal quantum simulators,” *Science*, vol. 273, no. 5278, pp. 1073–1078, 1996. DOI: 10.1126/science.273.5278.1073.
- [25] Naomichi Hatano and Masuo Suzuki, “Finding Exponential Product Formulas of Higher Orders,” in *Quantum Annealing and Other Optimization Methods*, vol. 679, Springer Berlin Heidelberg, 2005, pp. 37–68. DOI: 10.1007/11526216_2.
- [26] Andrew M. Childs *et al.*, “Theory of Trotter error with commutator scaling,” *Physical Review X*, vol. 11, no. 1, p. 011020, 2021. DOI: 10.1103/PhysRevX.11.011020.
- [27] D.J. Griffiths, *Introduction to Quantum Mechanics*. Pearson Prentice Hall, 2005.
- [28] W. Heisenberg, “Zur Theorie des Ferromagnetismus,” *de, Zeitschrift für Physik*, vol. 49, no. 9, pp. 619–636, 1928. DOI: 10.1007/BF01328601.
- [29] David Sherrington, “Neural networks: The spin glass approach,” en, in *North-Holland Mathematical Library*, vol. 51, Elsevier, 1993, pp. 261–291. DOI: 10.1016/S0924-6509(08)70040-0.
- [30] Roman Martonak, Giuseppe E. Santoro, and Erio Tosatti, “Quantum annealing of the Traveling Sales-

- man Problem,” *Physical Review E*, vol. 70, no. 5, p. 057701, 2004, arXiv:cond-mat/0402330. DOI: 10.1103/PhysRevE.70.057701.
- [31] Dominic W. Berry *et al.*, “Efficient Quantum Algorithms for Simulating Sparse Hamiltonians,” en, *Communications in Mathematical Physics*, vol. 270, no. 2, pp. 359–371, 2007. DOI: 10.1007/s00220-006-0150-x.
- [32] Pauli Virtanen *et al.*, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020. DOI: 10.1038/s41592-019-0686-2.
- [33] M. B. Hastings, “An Area Law for One Dimensional Quantum Systems,” *Journal of Statistical Mechanics: Theory and Experiment*, vol. 2007, no. 08, P08024–P08024, 2007, arXiv:0705.2024 [cond-mat, physics:math-ph, physics:quant-ph]. DOI: 10.1088/1742-5468/2007/08/P08024.
- [34] Guifré Vidal, “Efficient classical simulation of slightly entangled quantum computations,” *Phys. Rev. Lett.*, vol. 91, p. 147902, 14 2003. DOI: 10.1103/PhysRevLett.91.147902.
- [35] Reza Haghshenas, “Optimization schemes for unitary tensor-network circuit,” en, *Physical Review Research*, vol. 3, no. 2, p. 023148, 2021, arXiv:2009.02606 [cond-mat, physics:quant-ph]. DOI: 10.1103/PhysRevResearch.3.023148.
- [36] Frank Schindler and Adam S. Jermyn, “Algorithms for tensor network contraction ordering,” en, *Machine Learning: Science and Technology*, vol. 1, no. 3, p. 035001, 2020, Publisher: IOP Publishing. DOI: 10.1088/2632-2153/ab94c5.
- [37] Eli A. Meiron *et al.*, *Optimizing Tensor Network Contraction Using Reinforcement Learning*, arXiv:2204.09052 [quant-ph], 2022. DOI: 10.48550/arXiv.2204.09052.
- [38] Ulrich Schollwöck, “The density-matrix renormalization group in the age of matrix product states,” *Annals of Physics*, vol. 326, no. 1, pp. 96–192, 2011. DOI: 10.1016/j.aop.2010.09.012.
- [39] Jutho Haegeman *et al.*, “Time-dependent variational principle for quantum lattices,” *Physical Review Letters*, vol. 107, no. 7, p. 070601, 2011. DOI: 10.1103/PhysRevLett.107.070601.
- [40] Sheng-Hsuan Lin and Frank Pollmann, “Scaling of neural-network quantum states for time evolution,” *Physica Status Solidi (b)*, vol. 259, no. 5, p. 2100172, 2022. DOI: 10.1002/pssb.202100172.
- [41] Philipp Niemann *et al.*, “QMDDs: Efficient quantum function representation and manipulation,” *IEEE Trans. on CAD of Integrated Circuits and Systems*, 2016.
- [42] Stefan Hillmich, Igor L. Markov, and Robert Wille, “Just like the real thing: Fast weak simulation of quantum computation,” in *Design Automation Conf.*, 2020.
- [43] Robert Wille, Lukas Burgholzer, and Michael Artner, “Visualizing decision diagrams for quantum computing,” in *Design, Automation and Test in Europe*, 2021.
- [44] Johannes Hauschild and Frank Pollmann, “Efficient numerical simulations with Tensor Networks: Tensor Network Python (TeNPy),” *SciPost Phys. Lect. Notes*, p. 5, 2018, Code available from <https://github.com/tenpy/tenpy>. DOI: 10.21468/SciPostPhysLectNotes.5. arXiv: 1805.00055.