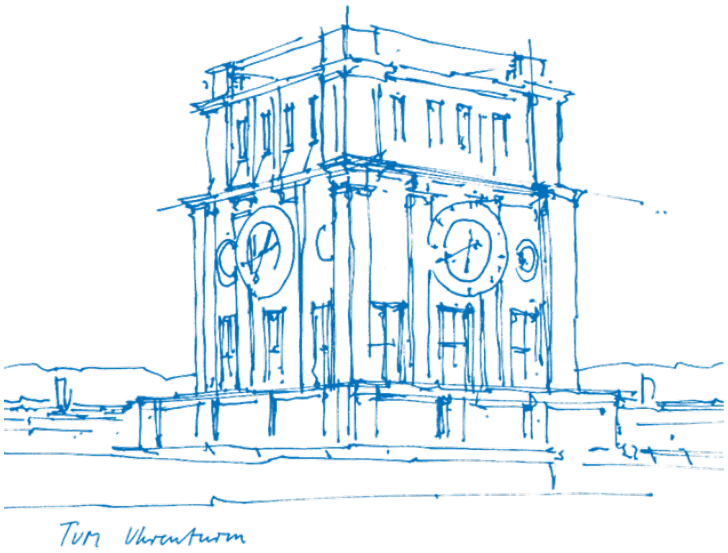
TUM School of Computation, Information and Technology
Technical University of Munich

MQT Qudits

Simulation and Compilation for Mixed-Dimensional Quantum Computing

Kevin Mato



MQT Qudits

Simulation and Compilation
for Mixed-Dimensional Quantum Computing

Kevin Mato

TUM School of Computation, Information and Technology
Technische Universität München

MQT Qudits

Simulation and Compilation for Mixed-Dimensional Quantum Computing

Kevin Mato

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitz:
Prof. Dr. Christian Mendl

Prüfer der Dissertation:
1. Prof. Dr.-Ing. Robert Wille
2. Prof. Dr. Philipp Hauke

Die Dissertation wurde am 05.06.2025 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 21.10.2025 angenommen.

Dedicated to my family and friends.

Acknowledgment

I express my sincere gratitude to all those who have helped and supported me personally and professionally throughout my journey toward this dissertation.

First, I want to thank my supervisor, Robert Wille, for the incredible opportunities he has provided me during this PhD journey and for teaching me the art of research. I am also deeply grateful to Martin Ringbauer, my indirect supervisor, for deepening my love for physics, introducing me to the fascinating world of qudits, and guiding me along the way. I also thank Prof. Philipp Hauke for his time and effort as the second examiner of my dissertation.

I also want to thank all the fantastic colleagues and friends from the Innsbruck Qudit project and the NeQST project for making this experience so enriching and unforgettable. I feel privileged to have worked alongside such talented individuals.

My heartfelt thanks go to all my colleagues and the rest of the team at the chair for their help, insightful discussions, and unwavering support throughout these years, and also for removing obstacles and ensuring everything ran smoothly, allowing me to focus on my work.

I am also grateful to all the friends and colleagues I met at conferences, workshops, and research visits who made my PhD journey truly unforgettable. If you're still reading this paragraph, I'm most likely talking about you. A special thanks also goes to Riccardo Mengoni for being both a mentor and a friend during these years and for being one of the reasons I continued working in quantum computing.

I am especially grateful to my friends, particularly those in Munich–the people who cared for me, made sure I was happy during difficult times, and helped me maintain a balanced life as much as possible through singing, dancing, playing sports, cooking, traveling, doing acrobatics, or simply talking.

Lastly, the biggest thank you goes to my family, the reason I have been able to reach my goals and dreams in life so far, and the reason why nothing ever seems too difficult to achieve. Without your constant support, I would not be here today. I am especially grateful to my sister for being both an inspiration and my best friend.

Abstract

Quantum computing, first proposed by Richard P. Feynman in 1982, promises to solve problems beyond the capabilities of classical machines and transform fields such as chemistry, artificial intelligence, physics, and cryptography.

Companies and academic institutions have made significant progress toward practical applications in the past decade. Partial credit is due to the development of quantum software alongside hardware advancements. Despite these strides, most quantum hardware and software are designed for computations based on qubits–two-level systems–representing only a fraction of quantum computing's potential. The full power of quantum computations could lie in leveraging higher-dimensional units of information, or quantum digits–qudits–which offer a range of advantages. Mixed-dimensional quantum computing, which combines qubits and qudits, provides a new synergistic approach to exploiting the best capabilities of both.

However, the scarcity of quantum software for mixed-dimensional quantum computers leaves much of their potential untapped. A large portion of the quantum computation design process still relies on manual techniques, which are difficult to generalize from just qubits to circuits incorporating both qubits and qudits. These manual processes are prone to human error, lead to inefficiencies, and divert efforts from conceptualizing novel applications and discoveries.

Mixed-dimensional quantum computing requires software solutions to address circuit design challenges, ensuring algorithms run quickly, reliably, and scalably. These challenges parallel those encountered in classical computing. The solution may lie in leveraging innovations in programming languages, computer architectures, compilers, optimization theory, and electronic design automation–advancements that have shaped classical computing. The goal is to enable users to express solutions in high-level programming languages while abstracting away complex low-level decisions on mixed-dimensional quantum architectures.

This Ph.D. thesis lays the groundwork for developing state-of-the-art methods and software to unlock mixed-dimensional quantum computing's potential. It does so by introducing novel methodologies inspired by classical computing techniques. To this end, it focuses on two primary quantum software design tasks:

- Classical simulation of quantum circuits, and
- Compilation of mixed-dimensional quantum circuits.

These tasks leverage data structures, like decision diagrams, and methods such as heuristic search, alongside quantum information to provide efficient strategies for designing circuits with optimal trade-offs between runtime, quality, and reliability.

The advancements presented in this thesis are integrated into MQT Qudits, a software framework part of the Munich Quantum Toolkit. This flexible yet integrated programming environment enhances accessibility to mixed-dimensional quantum computing and transforms it from an academic concept into a fully operational system. Additionally, it serves as a common platform for scientific collaboration among different physics communities.

By addressing the key challenges in developing mixed-dimensional quantum software and providing open-source solutions, this Ph.D. thesis advances quantum computing and lays a foundation for its next generation, highlighting the transformative role of classical computing design methods in shaping its future.

Zusammenfassung

Die Quanteninformatik, erstmals 1982 von Richard P. Feynman vorgeschlagen, verspricht, Probleme zu lösen, die über die Fähigkeiten klassischer Maschinen hinausgehen, und Bereiche wie Chemie, künstliche Intelligenz, Physik und Kryptografie zu transformieren.

Unternehmen und akademische Einrichtungen haben in den letzten zehn Jahren erhebliche Fortschritte bei der praktischen Anwendung erzielt. Dies ist zum Teil auf die Entwicklung von Quantensoftware und Fortschritte in der Hardware zurückzuführen. Trotz dieser Errungenschaften sind die meisten Quantenhardware und Softwarelösungen für Berechnungen mit Qubits–Zweizustandssystemen–ausgelegt, wodurch nur ein Bruchteil des Potenzials der Quanteninformatik genutzt wird. Die volle Leistungsfähigkeit von Quantenberechnungen könnte in der Nutzung höherdimensionaler Informationseinheiten, sogenannter Qudits, liegen, die zahlreiche Vorteile bieten. Das gemischt-dimensionale Quantencomputing, das Qubits und Qudits kombiniert, eröffnet einen neuen synergetischen Ansatz zur Nutzung der besten Eigenschaften beider Systeme.

Jedoch bleibt aufgrund des Mangels an Quantensoftware für gemischt-dimensionale Quantencomputer ein Großteil ihres Potenzials ungenutzt. Der Entwurfsprozess von Quantencomputern basiert weiterhin auf manuellen Techniken, die sich nur schwer von Qubits auf Schaltungen mit sowohl Qubits als auch Qudits übertragen lassen. Diese manuellen Prozesse sind fehleranfällig, ineffizient und lenken Ressourcen von der Entwicklung neuer Anwendungen und Entdeckungen ab.

Das gemischt-dimensionale Quantencomputing erfordert Softwarelösungen, um die Herausforderungen beim Schaltungsentwurf zu bewältigen und sicherzustellen, dass Algorithmen schnell, zuverlässig und skalierbar ausgeführt werden. Diese Herausforderungen ähneln denen des klassischen Rechnens. Eine mögliche Lösung liegt in der Nutzung von Innovationen in Programmiersprachen, Computerarchitekturen, Compilern, Optimierungstheorie und elektronischer Designautomatisierung–Fortschritte, die auch das klassische Rechnen geprägt haben. Ziel ist es, Nutzern zu ermöglichen, Lösungen in fortschrittlichen Programmiersprachen auszudrücken, während komplexe Entscheidungen auf niedriger Ebene in gemischt-dimensionalen Quantenarchitekturen abstrahiert werden.

Diese Doktorarbeit legt den Grundstein für die Entwicklung modernster Methoden und Software, um das Potenzial des gemischt-dimensionalen Quantencomputings zu erschließen. Dies geschieht durch die Einführung neuartiger Methoden, inspiriert von klassischen Rechentechniken. Dabei konzentriert sie sich auf zwei zentrale Aufgaben im Entwurf von Quantensoftware:

- Klassische Simulation von Quantenschaltungen, und
- Kompilierung von gemischt-dimensionalen Quantenschaltungen.

Diese Aufgaben nutzen Datenstrukturen (z. B. Decision Diagrams) und Methoden wie heuristische Suche in Kombination mit Quanteninformationen, um effiziente Strategien für den Schaltungsentwurf mit optimalen Kompromissen zwischen Laufzeit, Qualität und Zuverlässigkeit bereitzustellen.

Die in dieser Arbeit vorgestellten Entwicklungen sind in MQT Qudits integriert, einem Software-Framework innerhalb des Munich Quantum Toolkit. Diese flexible, aber dennoch integrierte Programmierumgebung verbessert die Zugänglichkeit des gemischt-dimensionalen Quantencomputings und verwandelt es von einem akademischen Konzept in ein voll funktionsfähiges System. Darüber hinaus dient sie als gemeinsame Plattform für die wissenschaftliche Zusammenarbeit zwischen verschiedenen Physikgemeinschaften.

Durch die Auseinandersetzung mit zentralen Herausforderungen und die Bereitstellung von Open-Source-Lösungen treibt diese Doktorarbeit das Quantencomputing voran und legt den Grundstein für dessen nächste Generation, wobei die transformative Rolle klassischer Berechnungsmethoden in der Gestaltung seiner Zukunft hervorgehoben wird.

Contents

Part I

Introduction and Background

1 Introduction

The progress of humanity has always been tied to curiosity and the ability to gain knowledge. To this end, computers have played an important role as aid in accomplishing great discoveries, such as the discovery of new materials and drugs or understanding the physics of our reality. Unfortunately, these problems come with extremely hard computational challenges that could potentially slow down progress or even lead to plateaus in scientific advancement. Here lies the idea of developing a new type of computer that could use principles originating from quantum information and technologies that allow new quantum information processing, called *Quantum Computing* (QC) [1]. Quantum information is an evolution of the classical information theory developed by Shannon, which is combined with new inputs from quantum mechanics, a fundamental theory that describes the behavior of nature at and below the scale of atoms. This field is the foundation of all quantum physics, which includes quantum chemistry, quantum field theory, and combined with the algorithmic theory of quantum computing and quantum communications.

In 1982, Richard Feynman proposed the idea of using devices and computational models that would be inherently quantum mechanical to simulate quantum many-body systems [2]. Since then, teams of quantum information theorists and experimentalists have been working on developing quantum computers. The discoveries made along the way in mathematics and physics have been instrumental in the growth of other fields, such as biology, chemistry, medicine, machine learning, finance, and signal processing.

The difference between quantum computers and classical ones starts with the representation of information. In classical computers, the minimal unit of information is the *bit*, where it can be exclusively observed as in 0 or 1. Instead, quantum computers represent their minimal units of information in states that can be in a *superposition*, which means the single unit is in both states 0 and 1 at the same time, while in the case of multiple units of information, this superposition can be extended to a special form called *entangled*. This particular property of quantum bits or *qubits* allows for operations that are not possible on classical bits. The operations, similar to the classical technology, are called gates, and they require lots of engineering in terms of hardware, in the form of design and control, to perform their task reliably, with error rates that could allow for complex calculations.

Over the past decade, the field of quantum computing has made significant progress, driven not only by advancements in hardware but also by theoretical and software progress. The adoption of the technology has been facilitated by the cloud access provided by several companies, which has increased its usability for a broader community of developers and users. These combined efforts have enhanced performance, scalability, and applicability in hybrid quantum-classical and purely quantum algorithms. Although a long way remains to go, companies are working hard to implement the best qubits and exploit them for useful commercial applications, with continuous comparisons between different technologies implementing the best qubits. Superconducting qubits are the focus of IBM, Google, and IQM, while Quantinuum and AQT specialize in trapped ions. Xanadu advances photonic chips, whereas QuEra, Planqc, and Pasqal have demonstrated the competitiveness of neutral-atom platforms.

Companies and academic institutions have made significant progress in achieving their roadmaps and goals for practical applications, and partial credit is due to the software developed for quantum computers (in short, quantum software) alongside hardware developments. In fact, quantum software has become increasingly important each year, given its role in enabling users to design, implement, and execute quantum

applications on devices. Industry and academia have collaborated to develop software frameworks. Famous examples include:

- IBM's Qiskit [3]
- Munich Quantum Toolkit [4]
- Google's Cirq [5]
- Microsoft's Quantum Development Kit (QDK) [6]
- Nvidia's CUDA-Q [7]
- Xanadu's Pennylane [8]

All of these frameworks operate across the quantum computing stack with varying levels of integration and hardware access. Most of them, so far, share a fundamental limitation: they are primarily designed for quantum computations based on qubits–two-level systems. Unfortunately, two-level systems are a restriction of the original nature of the quantum technology platforms, which are *multi-level quantum systems*, and despite the impressive achievements of industry and academia, the progress based on qubits represents only a fraction of quantum computing's true potential. The full power of quantum computations could lie in leveraging the higher-dimensional units of information for calculations with quantum digits–*qudits*, which offer a wide range of advantages, including expanded gate sets, higher information density per unit of information, and improved computational efficiency.

Qudits might play a key role in overcoming not only the limitations of classical machines but also those of current qubit-based quantum devices. This new advancement in quantum architectures does not mean that qubit quantum computers are obsolete but rather offers a new native and synergetic possibility for mixing the best capabilities of both worlds, offering significant potential for tailored methods, error mitigation strategies, and optimal algorithm (or circuit) designs, allowing for a better codesign of quantum computers and applications. This is the advent of a new paradigm of quantum computing and applications that require the use of qubits and qudits with arbitrary dimensions, thus enabling what is termed *mixed-dimensional quantum computing*.

Recent experimental demonstrations [9] have shown that qudits enable improvements in circuit complexity and algorithmic efficiency across a wide range of problems, with competitive error rates to qubits and more naturally suited encodings for the applications. Mixed-dimensional quantum computing is full of revolutionizing promises. However, like its qubit-based counterpart, it faces challenges not only in experimental control but also in algorithm design and particularly in quantum software engineering, which is fundamentally more complex. On top of these challenges, there is a widespread lack of quantum software development for qudit systems that ensures that their large potential remains largely untapped.

The consequential scarcity of automated software solutions for mixed-dimensional systems means that a considerable portion of the design process of quantum computations still heavily relies on manual techniques. Several of these methods are also no longer straightforward to generalize from the qubit circuit case to the qudit circuit case or to circuits incorporating both qubits and qudits. Furthermore, as the size of the qudits in a circuit increases, the associated tasks become exponentially more difficult. Such manual processes are not only susceptible to human error, but also lead to inefficiencies. The absence of systematic, automatic tools poses a significant obstacle to the progress and reliability of mixed-dimensional quantum computers. Moreover, it reduces the potential for scientific breakthroughs by diverting workforce efforts toward implementation rather than conceptualizing novel applications and discoveries. The problems of the manual design of mixed-dimensional quantum computations are particularly evident when compared

with qubit-based quantum computing, which has started to benefit from advanced and well-established software tools that are widely implemented on larger systems.

It becomes clear that mixed-dimensional quantum computing requires software solutions to address design challenges for circuits that enable algorithms to run quickly, reliably, and scalably. This is crucial to prevent a scenario where mixed-dimensional quantum hardware becomes increasingly available, yet lacks reliable, scalable, and practical software solutions. Quantum software faces design challenges reminiscent of those previously encountered in classical computing. In classical systems, software advancements supported the design and development of modern circuits and systems by reducing development cost and time while enhancing performance and reliability. Achieving this required innovations in programming languages, computer architectures, compilers, optimization theory, and electronic design automation (EDA), which abstracted design complexities and allowed users to focus on programming their algorithms. Similarly, the envisioned flow of quantum software should parallel that of classical software, enabling users to express their solutions in high-level programming languages while abstracting away complex low-level decisions. To replicate this success in mixed-dimensional quantum computing, quantum software should provide features such as:

- Simulators as means of testing, verification, and validation of the circuit designs, that are based on user input.
- Compilers that could synthesize circuits by following user input and optimizing them according to the best practices to reduce the risk of errors and resource usage.

This Ph.D. thesis lays the groundwork for developing methods and state-of-the-art software that will unleash the potential of mixed-dimensional quantum computing. It concentrates on two primary design tasks within quantum software: simulation and compilation, with efficient novel methodologies inspired by fundamental techniques previously developed in traditional circuits and systems. Specifically, this involves leveraging data-structures (e.g., adapted decision diagrams) and methods, such as heuristic search alongside quantum information, to offer efficient strategies for simulating and compiling quantum computer circuits with optimal trade-offs between memory, runtime, quality, and reliability of the designed circuits.

This Ph.D. thesis is organized into three core parts, two of which will address specific tasks handled by quantum software, and a third one will present the resulting open-source framework developed to enable mixed-dimensional quantum computing. The following summarizes the key contributions of each part presented in this thesis.

Classical Simulation of Quantum Circuits The classical simulation of quantum circuits is crucial for various aspects of quantum application development. It allows for algorithm verification when access to appropriate hardware is restricted, offers full quantum state insight for troubleshooting, sets standards for assessing quantum hardware performance and validates the corresponding results. Despite its utility, simulating quantum systems with mixed dimensions poses specific difficulties due to the exponential expansion of the state space as both the number of qudits and their individual dimensions increase, rendering traditional qubit simulation techniques insufficient right off the bat. The basics for classical simulation of quantum circuits, the data-structures for improving the performance and memory consumption and the noise models for validating the results of mixed-dimensional quantum computers are explored in *Part* II, divided in two chapters:

***Chapter* 4, "Mixed-Dimensional Quantum Circuit Simulation with Decision Diagrams"** This chapter, based on [10], describes the first ad hoc classical simulator for mixed-dimensional qudit systems. To this end, we introduced a type of decision diagram capable of handling the simulation of mixed-dimensional

quantum circuits and presented a corresponding implementation. The achievement is the enhanced flexibility of decision diagrams and high-performing simulation time in the order of minutes for relatively complex calculations.

***Chapter* 5, "Refining and Simulating Mixed-Dimensional Quantum Noise"** This chapter advances the current field of classical quantum circuit simulation by providing the first formalization of noise models in mixed-dimensional systems, capable of approximating noise in both shallow and deep circuit regimes, by utilizing established qubit theory and simulation techniques, with potential advantages for decision diagram simulation. The resulting versatile model allows for realistic simulations of mixed-dimensional quantum circuits with close adherence to the experiments, thanks to the implementation that leverages stochastic (Monte-Carlo) simulations and allows for likely faster memory access in the case of DD simulations. The first result of this type for mixed-dimensional quantum computers.

Compilation of Quantum Circuits Like in classical computers, compiling remains a critical challenge. In classical systems, it involves the conversion of human-readable text into assembly instructions. However, in quantum systems, it involves translating abstract quantum algorithms into sequences of operations that can be physically realized, which is complex in qubit devices and grows considerably more intricate in mixed-dimensional machines. The compilation process must consider varying dimensions, diverse operation types, and hardware-specific constraints while striving to optimize both the depth of the circuit, its resistance to errors, and optimizing classical runtime and memory consumption. The complexity of compiling is further worsened by the requirement to manage local operations on individual qudits, which will introduce new constraints compared to the qubit case, as well as by the possibility of applying entangling operations between qudits of differing dimensions; *Part* III will explore the problems in more detail.

This Ph.D. thesis lays the groundwork for compiling mixed-dimensional quantum computers, focusing on establishing foundational principles, methodologies and data-structures. It examines quantum circuit compression, a process that reduces the number of operations in existing qubit circuits by leveraging mixed-dimensional architectures more effectively. Additionally, it explores the synthesis of circuits generating specific quantum states and the decomposition of abstract quantum operations. The aim is to find the shortest possible sequence of arbitrary quantum operations to implement a given quantum state or operation. The following chapters of *Part* III explore one by one these different tasks encountered in compilation, more in detail:

***Chapter* 8, "Compression of Qubit Circuits: Mapping to Mixed-Dimensional Quantum Systems"** This chapter, based on [11], presents a method for compressing, or converting, qubit circuits into qudit circuits. This transformation allows algorithms originally designed for qubits to be restructured and mapped on hardware that utilize higher-dimensional qudits and that is tailored to the circuit. This compression reduces the number of non-local operations–minimizing unwanted noise in quantum computations. The positive results show the effectiveness of the method and indicate a potential secondary use of the method and the relative tool, as a resource estimator technique to investigate mixed-dimensional architectures and enhance the execution of qubit circuits.

***Chapter* 9, "Mixed-Dimensional Qudit State Preparation Using Edge-Weighted Decision Diagrams"** This chapter, based on [12], introduces an innovative synthesis approach for quantum circuits that target arbitrary states within mixed-dimensional quantum systems. The method utilizes edge-weighted decision diagrams with a variable number of successors for state representation and quantum circuit synthesis. By employing advanced approximation strategies on the data-structure, the gate count is significantly

optimized in the circuits. This method proves highly efficient and versatile for the preparation of arbitrary quantum states in any mixed-dimensional quantum architecture, representing the first automated approach applicable to quantum architectures comprising qubits and qudits of varying dimensions.

The second half of this part of the thesis lays a solid foundation for automated decompositions that are resilient to noise, compatible with qudit structures, and aligned with the hardware.

***Chapter* 10, "Compilation of Entangling Gates for High-Dimensional Quantum Systems"** This chapter, based on [13], introduces a universal workflow for compiling any two-qudit unitaries into any hardware gate set supporting two-level entangling gates. While current qudit entangling gates require manual, system-specific implementation, the method offers an algorithmically efficient approach, with pre-computable numerical optimization. Although the gate-count is not optimal, case studies demonstrate the workflow's practical feasibility across different quantum systems, overcoming the difficulties of compiling two-qudit entangling gates on a mixed-dimensional architectures.

***Chapter* 11, "Adaptive Compilation of Multi-Level Quantum Operations"** This chapter, based on [14], presents an algorithm aimed at efficiently breaking down local unitaries in multi-level quantum systems. This algorithm optimizes the mapping and routing of logical states to energy levels for better compilation efficiency and provides a new function to assess the decomposition cost. Since common decomposition algorithms return sequences with lots of redundancies, the new adaptable methods enhances gate costs while also taking into account realistic coupling constraints and rotation expenses.

Open Source Software and Languages This final part of this Ph.D. thesis, based on [15], demonstrates how the simulation and compilation results, introduced in earlier parts, provide powerful tools individually and reach their true potential when integrated into a cohesive framework. The developments presented in this Ph.D. thesis have been condensed into a software framework named *MQT Qudits*, which is part of the *Munich Quantum Toolkit [4]*. This framework is described in *Part* IV, which demonstrates how users can interact with mixed-dimensional quantum computers through an innovative assembly language and compares this with the current landscape of quantum computing languages. This integration introduces a unified control, transforming separate components into a comprehensive quantum software stack. Users can also express, simulate, and compile quantum algorithms through a unified Python interface, an approach particularly significant for mixed-dimensional quantum computing. This flexible yet integrated programming environment maximizes the accessibility to mixed-dimensional quantum computing and advances it from academic theory to a fully operational system, while becoming a common ground for scientific communication between different physics communities.

To ensure that this thesis is self-sufficient and well contextualized, *Chapter* 2 provides the essential background on quantum computing. It also reviews key data-structures and the principles of quantum information and computer science that are essential to comprehend the methodologies at the base of this Ph.D. thesis. Through this approach, the thesis offers a comprehensive guide and framework for developing new automation methods and software suitable for mixed-dimensional quantum computing.

Publications and Code Availabilty The methods developed in this Ph.D. thesis have been thoroughly validated through peer review and published in major design automation conferences (such as DAC and ASPDAC), as well as quantum venues (such as QCE, QSW). The findings presented in this thesis are derived as follows: *Chapter* 4 builds on the simulation work from [10]. *Chapter* 8 addresses the compression of qubit circuits, based on [11]. *Chapter* 9 enhances state-of-the-art synthesis for state

preparation circuits described in [12]. Meanwhile, *Chapter* 10 refers to [13], and *Chapter* 11 illustrates the theory and techniques in [14]. In *Part* IV, the chapter concerning open-source software and languages is based on [15]. Furthermore, we emphasize that all methods and tools have been released as open-source software, facilitating reproducibility and wider adoption among researchers and practitioners.

Further Accomplishments In addition to the achievements described above, the following additional accomplishments have been achieved within the context of this thesis:

- Several top placements in various quantum computing IBM challenges and summer schools
- Multiple recognitions from institutions and the community:
  - EU Innovation Radar Nomination
  - The Oak Ridge National Laboratory Best Poster Award
  - Unitary Fund Grant Winner
- Numerous invited seminars and lectures, some of the most relevant are the following:
  - Invited lecture on “Introduction to Quantum Computing” at the European Summer School of the Italian Supercomputing Center CINECA
  - Invited seminar on "Software for Mixed-Dimensional Quantum Computing" at the University of Oxford
  - Invited seminar on "Design Automation for Quantum Comuting" at the Barcelona Supercomputing Centre (BSC-CNS)
  - Invited talk at the Workshop on Quantum Software, co-located with QTML22

2 Background

This chapter provides an overview of the most fundamental concepts of quantum computing, especially in the context of higher-mixed-dimensional quantum systems, and of some of the fundamental data structures and methods based on computer science that will ensure that the thesis is self-contained. This chapter will begin by building up the understanding of the basics of quantum computing in a bottom-up fashion, where first we will review the basic mechanisms of quantum hardware that allow us to store and manipulate information on a quantum computer, and afterwards we will illustrate the concepts of a quantum state, operations, circuits, and applications that can benefit from mixed-dimensional quantum systems, and the rest of the material will concisely describe the aspects relevant to the subsequent chapters. Additional details and information will be provided in subsequent sections. For those interested in learning further about quantum computing and high-dimensional quantum computing, references [1], [16] are recommended.

2.1 Platforms for Quantum Computing

Before exploring the core properties of quantum computing and how quantum information is manipulated to achieve meaningful computations, it is useful to understand where and how this information is processed. It is common to talk about devices that can operate exclusively with two-level systems–qubits. However, the hardware natively supports a wider choice of quantum information processing that allows encoding multi-valued logic in higher-dimensional units of information, called *quantum digits* (qudits), enabled by the inherent multi-level architecture of quantum systems.

Quantum information processing is made possible by the way energy is structured in the microscopic world. In contrast to classical systems, where energy can assume any value, quantum systems possess distinct, quantized energy levels. This quantization forms the foundational structure for quantum information processing. Specifically, each energy level corresponds to a particular state of the physical system, which aligns with one of the logical states of our logic encoding. Consequently, to alter the state of a quantum system, one must change its energy.

More in detail, identifying the energy levels of a quantum system requires using the quantum mechanical description of the system, which begins with the time-independent form of the Schrödinger equation

$$H|\psi_n\rangle = E_n|\psi_n\rangle \ , \tag{2.1}$$

where H represents the system's total energy operator (a descriptor called "Hamiltonian"), $|\psi_n\rangle$ that represents all the possible states of the system, and E_n represents the corresponding energy value of a quantized state $|\psi_n\rangle$.

Example 1. *Trapped ion and superconducting qubits exhibit varied modalities in quantum control, affecting the way their energy is manipulated. The example illustrates the physical variables that define the energy, and consequently the Hamiltonian of a system, for two cases:*

- *In trapped ions, information is encoded in the state of the free electron, with the Hamiltonian given by*

$$H = \omega\sigma_z \tag{2.2}$$

where ω is the energy difference between the two states of the electron (which corresponds to the frequency of the transition between these states), and σ_z is the Pauli z operator, which acts on the two-level system (the electron), with eigenvalues ± 1.

- *Superconducting qubits, also called artificial atoms, create quantum states through the interplay of charge and magnetic flux. In practice, superconducting qubits are resonant circuits (harmonic oscillators) with a nonlinear element in the Josephson junction. This nonlinearity turns it into an anharmonic oscillator, which makes it possible to use as a qubit or qudit. The Hamiltonian is described by*

$$H = 4E_C(n - n_g)^2 - E_J \cos(\phi) \ , \tag{2.3}$$

where E_C is the charging energy and E_J is the Josephson energy. Unlike natural atoms, these energy levels can be engineered by circuit design.

Quantum information processing platforms operate on the energy states through an evolution, which is better described by the time-dependent Schrödinger equation

$$i\hbar \frac{\partial}{\partial t}|\psi(t)\rangle = H|\psi(t)\rangle \ , \tag{2.4}$$

by following this formula and a small rewriting, we notice that it describes how these states evolve through unitary evolution

$$U(t) = e^{-iHt/\hbar} \ , \tag{2.5}$$

These properties collectively form the theoretical backbone for quantum information manipulation, where discrete energy levels act as the computational basis states, unitary evolution describes quantum operations, and coherent control via time-dependent Hamiltonians $H = H_0 + H_{\text{int}}(t)$ (H_0 is the initial Hamiltonian and H_{int} stays for interval) allows for accurate state manipulation and evolution. Since the Hamiltonian is a Hermitian matrix, selecting a fixed evolution time t ensures that the exponential matrix results in a unitary matrix. Therefore, by varying Hamiltonians and evolution times, it becomes possible to create operations or *quantum gates*. A unitary evolution, combined with the superposition principle of quantum states, that will be later on better described, can be written as:

$$|\psi(t)\rangle = \sum_n c_n e^{-iE_n t/\hbar}|\psi_n\rangle \tag{2.6}$$

which forms the basis for quantum information processing, where each energy eigenstate evolves with its own characteristic phase factor induced by the unitary evolution, enabling a form of parallelism. The equation encodes multiple fundamental aspects of quantum mechanics: the preservation of probability through unitary evolution, the emergence of quantum bits through differences in energy levels, and the relationship between energy and frequency through $E = \hbar\omega$. For more details, we point the reader more interested in quantum mechanics to reference [17].

Example 2. *In practice, how we shift from physical energy level interpretation to quantum information storage occurs through the following mechanism divided into three steps: the first step is the state identification, where we map physical energy levels to computational basis states, for example, in a three energy levels system:*

- $|0\rangle \leftrightarrow$ *ground state*
- $|1\rangle \leftrightarrow$ *first excited state*
- $|2\rangle \leftrightarrow$ *second excited state*

In this way, the physical system that has three different energy levels, can finally describe an entity with three logic states.

The second step consists in performing the state encoding, which by going through several unitary evolutions, will be in a superposition of this from: $|\psi\rangle = \sum_{d=0}^{2} c_d|d\rangle$*. Lastly, we evolve the state*

$$|\psi(t)\rangle = 0.1 * e^{-i0.9*t/\hbar}|0\rangle + 0.3 * e^{-i0.001*t/\hbar}|1\rangle + 0.2 * e^{-i0.7*t/\hbar}|2\rangle$$

depending on the amount of time the system is made evolve, the final state will be different. In Section 2.2.1, we will discuss the properties of quantum states.

The principles of quantum information processing described so far are valid for all quantum technology platforms, therefore various technological platforms have been examples of proof-of-concept NISQ [18] devices, such as superconducting circuits [19], trapped ions [20], and single photons [21]. Proof-of-principle experiments [9], [22]–[24] have demonstrated that qudits facilitate reductions in circuit complexity and boost algorithmic efficiency across numerous problem categories. These findings have produced proposals and the realization of basic qudit control across various physical platforms, including trapped ions [9], [25], photonic systems [22], [26]–[28], superconducting circuits [29], [30], Rydberg atoms [31], [32], nuclear spins [33], cold atoms [34], [35], nuclear magnetic resonance systems [23] and molecular spin [36]. Recently, efforts have intensified, leading to the demonstration of a universal qudit quantum processors achieving error rates comparable to qubit-based systems [9], [37]. In the past, all these developments in quantum control have naturally led to studies on the computational applications of qudits, with research focusing primarily on early theoretical analyses of algorithms for ideal qudits, in comparison to qubits, for example in [16]. Given the numerous experimental demonstrations we have just seen, that some of them have are being used for practical applications [9], and the stronger theoretical interest in this type of technology platform, we can say that mixed-dimensional quantum computing is a reality.

2.2 Fundamentals of Quantum Computing

The quantum hardware mechanisms we have just examined will serve as the foundational layer of abstraction for our computations, forming the basis on which we will develop our understanding in this section. We have introduced the notion of multi-level quantum systems and explained how information units are formed during the state identification phase. In light of the increased flexibility gained by incorporating qudits of varying dimensions into our architectures, this section presents the core concepts of mixed-dimensional quantum computing, while providing comparisons between qubits and qudits when appropriate.

2.2.1 Quantum States

Classical computing systems are built on a model that uses bits, which can only be 0 or 1 upon observation. In previous sections, it was noted that quantum systems can enable multi-level logic, where the fundamental units of information are *qubits* and *qudits*. The key distinction between classical and quantum computing stems from the fact that qubits and qudits can represent any linear combination of quantum states from $|0\rangle$ to $|d-1\rangle$, expressed through Dirac's bra-ket notation [1], known as *superposition*. The quantum state can thus be represented as

$$|\psi\rangle = \alpha_0 \cdot |0\rangle + \alpha_1 \cdot |1\rangle + \ldots + \alpha_{d-1} \cdot |d-1\rangle \ , \tag{2.7}$$

where $\alpha_i \in \mathbb{C}$ signify the amplitudes associated with the orthonormal basis vectors $|0\rangle, |1\rangle, |2\rangle, ..., |d-1\rangle$ of the Hilbert space. Alternatively, the state may be viewed as a vector within the d-dimensional Hilbert space $\mathcal{H}_d$, i.e.

$$|\psi\rangle = \left[\, \alpha_0 \ \alpha_1 \ \ldots \ \alpha_{d-1} \,\right]^{\mathrm{T}} . \tag{2.8}$$

The probability of observing the corresponding basis state i when measuring the qudit is represented by the square of the magnitude of each amplitude, $|\alpha_i|^2$. These amplitudes must satisfy the condition $\sum_{i=0}^{d-1} |\alpha_i|^2 = 1$, as the probabilities must sum to 1.

When dealing with multiple qubits and qudits in a systems S, with D units, the simplest scenario involves forming the tensor product of individual states, i.e.

$$|\psi_S\rangle = |\psi_0\rangle \otimes |\psi_0\rangle \cdots \otimes |\psi_{D-1}\rangle \ . \tag{2.9}$$

Here, the final state remains a linear combination of tensor products of the basis states of each unit, with d denoting the size of each unit, i.e.

$$|\psi\rangle = \alpha_{0\ldots0} \cdot |0 \ldots 0\rangle + \ldots + \alpha_{d_0-1\ldots d_{(D-1)}-1} \cdot |d_0 - 1 \ldots d_{(D-1)} - 1\rangle \ . \tag{2.10}$$

However, the tensor product scenario is straightforward, as quantum states can exhibit *entanglement*, arising from interactions in systems with multiple qudits. Entanglement is a powerful form of quantum correlation, where the information is infused in the state's entirety and can no longer be extracted from the individual qudits. Moreover, entanglement becomes a big complexity when studying systems that are made of multiple qubits, or qubits and qudits. In fact, there are different types of entanglement, especially in mixed-dimensional systems. It will be discussed in Section 2.2.2 and in Example 4 there is a first hint.

Example 3. *Consider a system of one qudit with only three energy levels (also referred to as* qutrit*). The quantum state* $|\psi\rangle = \sqrt{1/3} \cdot |0\rangle + \sqrt{1/3} \cdot |1\rangle + \sqrt{1/3} \cdot |2\rangle$ *is a valid state with equal probability of measuring each basis. Equivalently, the quantum state may be represented as vector* $\sqrt{1/3} \cdot \begin{bmatrix} 1 & 1 & 1 \end{bmatrix}^{\mathrm{T}}$*.*

In a similar fashion, quantum systems of mixed dimensions *can be constructed. Extending the previous qutrit state by a qubit enables the representation of the state:*

$$|\psi'\rangle = \sqrt{1/3} \cdot |0\rangle|0\rangle + \sqrt{1/3} \cdot |1\rangle|1\rangle + \sqrt{1/3} \cdot |2\rangle|0\rangle \ ,$$

equivalently represented by the vector $\sqrt{1/3} \cdot \begin{bmatrix} 1 & 0 & 0 & 1 & 1 & 0 \end{bmatrix}^{\mathrm{T}}$

Example 4. *Consider a composite system of a qubit (A) and a qubit (B). An example of a maximally entangled state between them is*

$$|\psi\rangle = \frac{1}{\sqrt{2}}(|0\rangle_A|0\rangle_B + |1\rangle_A|1\rangle_B) \tag{2.11}$$

Now let us extend qubit B to a qutrit, a three-level system. Note that even though the qutrit has access to state $|2\rangle_B$*, the particular entangled state in Eq. (2.11) only uses two of its levels. This state cannot be written as a product state* $|\psi\rangle_A \otimes |\phi\rangle_B$*, demonstrating genuine quantum entanglement. In particular, this state exhibits asymmetric entanglement between the qubit and qutrit systems. Finally, if we expanded qubit A to a qutrit we could generate a more general entangled state using all the qutrit levels, which could be:*

$$|\psi\rangle = \frac{1}{\sqrt{3}}(|0\rangle_A|0\rangle_B + |1\rangle_A|1\rangle_B + |2\rangle_A|2\rangle_B) \tag{2.12}$$

2.2.2 Quantum Operations

As discussed earlier in Section 2.1, the state of a single d-level qudit system can be modified by transformations symbolized by $d \times d$ unitary matrices $U \in \mathrm{SU}(d)$. The matrices U transform complex vectors into other complex vectors, and they must satisfy the condition $U^\dagger U = UU^\dagger = I$. The post-operation state is achieved by left-multiplying the matrix U with the initial state, like in Example 5.

Example 5. *Consider a three-level qudit (i.e., a qutrit) initially in the state* $|0\rangle$*. Applying the Hadamard operation* H *to it yields the output state shown before in Example 3, i.e.,*

$$H \cdot |0\rangle = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 & 1 & 1 \\ 1 & e^{\frac{2\pi i}{3}} & e^{\frac{-2\pi i}{3}} \\ 1 & e^{\frac{-2\pi i}{3}} & e^{\frac{2\pi i}{3}} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \frac{1}{\sqrt{3}} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}.$$

Quantum operations can be either local or non-local–entangling operations, can operate on the whole qudit space, or on subspaces of it. Each operation has some mathematical properties that are defined by mathematical groups like the Pauli group [38], or Clifford [39], or non-Clifford group [40]. Knowing to which group an operation belongs allows us to understand its effect in a sequence, its capabilities in expressing an algorithm, and how the single operation should be potentially translated into instructions compatible with the quantum computer.

Examples of local quantum operations are the Pauli gates

$$X_2 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \qquad Z_2 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \tag{2.13}$$

In this case, 2x2 matrices that correspond to a bit flip and a phase flip on a qubit. The generalization of the qubit Pauli is the *clock* and *shift* operators. In the case of a qutrit, they are described as

$$X_3 = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \quad Z_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \omega & 0 \\ 0 & 0 & \omega^2 \end{bmatrix}, \tag{2.14}$$

with $\omega = e^{\frac{2\pi i}{d}}$ and d being the dimension of the single qudit. These operators are basically the arithmetic "plus one" in modulo d dimension of the qudit and a phase shift for each dimension of the qudit. Something really important is that by composing these operations, we can form the qudit Pauli group, as

$$X^{\alpha} \cdot Z^{\beta}, \quad \alpha, \beta \in [0, d-1]\,. \tag{2.15}$$

The characteristics of the qudit Pauli group [41] operations are considerably more intricate than those of the qubit Pauli group. For instance, applying a qudit operation twice does not always result in the identity effect. Together with the Pauli group, we define the Clifford group. A Clifford operation refers to any unitary transformation that maintains the Pauli group when conjugated–meaning if you conjugate any Pauli operator by a Clifford, you get another Pauli operator. In other words, a Clifford C has the property that for any Pauli P, $CPC^{\dagger}$ is another Pauli operator. Another intriguing feature of qudit systems is their ability to exceed these two categories. Operations on single qudits can be composed to focus on particular subspaces or the qudit's entire Hilbert space. For example, subspace operations might include rotations constructed from Gell-Mann matrices [9], which physically represent two-level rotations within certain qudit subspaces, like Example 6.

Example 6. *This instance illustrates how the qubit bit-flip operation is embedded in a three-dimensional qudit or qutrit space. Specifically, the subspace* $|0\rangle$*,* $|1\rangle$ *is the focus of manipulation, since the rotation interchanges these two dimensions within the three-dimensional framework.*

$$R_{0,1} = \begin{array}{c|ccc} & |0\rangle & |1\rangle & |2\rangle \\ \hline |0\rangle & 0 & 1 & 0 \\ |1\rangle & 1 & 0 & 0 \\ |2\rangle| & 0 & 0 & 1 \end{array} \tag{2.16}$$

This is clearly a rotation within a subspace as it involves two dimensions, with the third ($|2\rangle$) remaining unchanged; however, we could have chosen the subspace $|0\rangle$, $|2\rangle$ as well. The possible choices for subspace operations are $(d(d-1))/2$ within a d-dimensional qudit. Note that it is important to recognize that inserting a subspace rotation into a higher-dimensional space can alter the characteristics of the operation; it may no longer belong to the Pauli group of the original dimension, or it could cease to be a Clifford. The conclusion is that naive generalization are not possible.

The power of quantum algorithms lies in the efficient manipulation of superpositions of quantum states by exploiting interference effects and the phenomenon of entanglement. Since qudit systems support a varied set of non-local operations their potential algorithmic efficiency is higher. Unfortunately, these operations cannot be simply obtained by generalizing entangling qubit operations. Additionally, realizing entangling operations in qubit systems is comparatively simple and rather unclear in qudit systems.

More precisely, for qubit systems it is sufficient to compile the CNOT gate to the native operations of the quantum hardware, because all qubit entangling operations can be implemented by the controlled-NOT (CNOT) gate and appropriate local operations on the subsystems [1]. In contrast, for qudit systems this does not hold anymore and entanglement can be generated in many inequivalent ways. Consequently, while any single entangling gate is sufficient for universal quantum computation [42], not all entangling gates are equally useful for any given application. On the high level, CNOT applies an X gate to the target qubit, whenever the control qubit is in the $|1\rangle$ state. The qudit-embedded version of the CNOT gate, called the Controlled-Exchange gate (CEX [9]), generates qubit-level entanglement in a high-dimensional Hilbert space, and similarly flips the subspace $|0\rangle$, $|1\rangle$, when the control qudit is in the control state.

Example 7. *The CNOT between two qubits and a CEX between a qubit and a qutrit have some similarities in terms of matrices*

$$\mathit{CNOT} = \left[\begin{array}{cc|cc} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ \hline 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{array}\right] \qquad \mathit{CEX} = \left[\begin{array}{ccc|ccc} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \tag{2.17}$$

In a more schematic fashion the behaviour of the CEX gate can be expressed as

$$\mathrm{CEX}_{c,t_1,t_2} : \begin{cases} \textit{Swap}(t_1, t_2) & \textit{if control is } c \\ \textit{Identity} & \textit{otherwise} \end{cases}, \tag{2.18}$$

where the variable c is the control state, and t_1 and t_2 are the target states. The CEX gate does not necessarily maintain the properties of the CNOT. In fact, the direct generalization the CNOT is the CSUM gate that directly generates qudit entanglement and defined by

$$\mathrm{CSUM} : |c, j\rangle \mapsto |i, i \oplus j\rangle, \tag{2.19}$$

where $\oplus$ denotes addition modulo the dimension d, which generates a complex and powerful d-leveled type of entanglement.

These are just two examples of a more general theme in qudit systems, where entangling gates differ in their entangling power*, and why non-local qudit gates can produce more entanglement than qubit ones are presented in depth in Ref. [43].*

2.2.3 Quantum Circuit

After discussing how states and operations are represented for qudits and mixed-dimensional systems, it is relevant to discuss how to apply quantum operations, called gates, in the quantum circuit representation and how algorithms for quantum systems can be represented as well.

Quantum computations evolve the state of a quantum systems, or the quantum computer, one operations at a time. At the current stage of quantum computing these sequences of operations are described by quantum circuits. Since a gate represents each operation, the quantum circuits C is a sequence of gates, that is read left to right:

$$C : G_0 \dots G_{|C|-1} \tag{2.20}$$

If we think of the circuit as a sequence of gates, then $|C|$ is the cardinality of the ensemble. Since we can associate a unitary matrix to each gate, then we can associate a unitary matrix also with the behaviour of the whole circuit.

$$U_C = U_{|C|-1} \cdot \dots \cdot \dots \cdot U_1 \cdot U_0 \tag{2.21}$$

For a quantum circuit consisting of multiple mixed-dimensional qudits, the unitary matrix will be of dimension $\prod_i d_i \times \prod_i d_i$ with d_i denoting the dimension of each qudit. The application of the operation will require an input state of the size of $\prod_i d_i$ entries. Later on, adjustments to the matrices to match the size of the system are going to be discussed in more depth.

So our U_C will usually be a pretty large matrix, since for each operation we will have to perform a sort of broadcasting operation as such, if it is a circuit of four qudits

$$U_{2inC} = I_{d_0} \otimes I_{d_1} \otimes U_2 \otimes I_{d_3} \tag{2.22}$$

where U_{2inC} is the unitary of the gate applied to qudit 2 with the appropriate size, and I_{d_i} is the identity matrix of the i-th qudit.

More in particular, the evolution of the state will be:

$$\begin{aligned} |\psi_1\rangle &= [I_{d_0} \otimes I_{d_1} \otimes U_2 \otimes I_{d_3}] \cdot \psi_0, \\ |\psi_2\rangle &= [I_{d_0} \otimes U_1 \otimes I_{d_2} \otimes I_{d_3}] \cdot \psi_1, \\ |\psi_3\rangle &= [I_{d_0} \otimes I_{d_1} \otimes U_{cex}] \cdot \psi_2, \\ &\vdots \\ |\psi_{final}\rangle &= [U_0 \otimes I_{d_1} \otimes I_{d_2} \otimes I_{d_3}] \cdot \psi_x \end{aligned} \tag{2.23}$$

Note that in the equation above the unitary of CEX takes the size of the two last qudits multiplied ($d_2 * d_3$, that is why I_3 is missing). This task is mimicked on a classical computer with classical simulation techniques and explained more in detail in *Part* II.

Quantum circuits have properties like circuit depth that depend on the number of entangling gates or layers of parallel gates in the sequence and have an approximation power, given by the universality of the gate-set [44]–[46] used and by the properties of each gate and capabilities in approximation unitary designs [47]–[49]. More will be discussed in *Part* III.

Example 8. *The quantum circuit in Figure 2.1 shows three lines, in order: a qubit (q0), a qutrit (q1) (3 levels), and a ququart (q2) (4 levels), as well as three different types of operations, a local operation, two controlled operations, and measurements. The controlled operations on qudits apply a unitary to a target line only if the control qudit is in a $|i\rangle$ between 0 and $d-1$, with d dimensions of the control qudit. The*

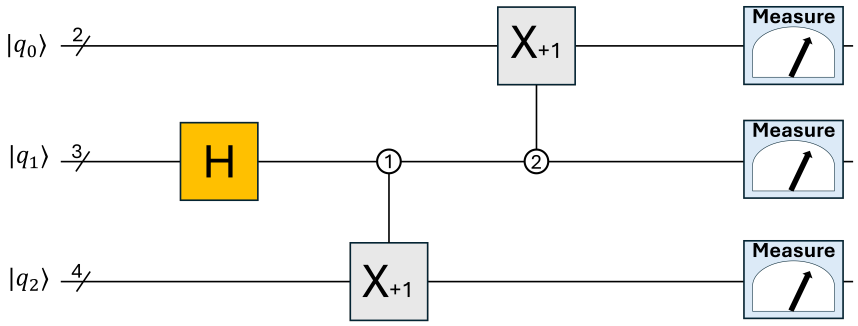


Figure 2.1 An example of a mixed-dimensional circuit with three qudits.

control state is marked inside the circle on the control line. The first operation is a Hadamard applied to the qutrit (q1). Afterwards a controlled-on-1 X is applied to the ququart (q2) and lastly to the qubit (normally addressed as CNOT*). Finally, the three units of information are measured in order to derive the outcomes of the circuit. The state of the system after the application of each operation can be viewed as* intermediate*, as the output state of one is the input state of another.*

2.2.4 Applications of Mixed-Dimensional Quantum Computing

After reviewing the concept of quantum circuits, in this section, we illustrate what kind of unique opportunities can mixed-dimensional quantum systems offer to advance quantum computations, and which quantum computing applications can benefit the most from the new architectures. Established quantum algorithms, such as Shor's algorithm for integer factorization [50], Grover's algorithm for unstructured data search [51], and developments in quantum chemistry [52], have focused primarily on two-dimensional–qubits systems, leaving the potential of the higher dimensions underutilized. Exploring these higher-dimensional systems is crucial for identifying opportunities of fruitful applications of quantum technologies in various contexts, especially when it comes to solving complex computational problems that remain intractable for classical computers.

Architectural Advantages for Qubit Circuits The new architectures come with unseen opportunities of all sorts. In more detail, mixed-dimensional systems can be leveraged as a platform to perform better computations of qubit-based logic. In fact, they offer several architectural benefits that fundamentally enhance the performance and efficiency of qubit quantum circuits, either by better exploitation of the hardware or by using better encodings or mappings, which show a remarkable improvement in multiple quality metrics.

In fact, it is possible to improve the gate decompositions of multi-qubit gates, for example, through the temporary expansion of the Hilbert space of one of the qubits into higher-dimensional units, leading to more compact circuits with reduced noise accumulation [53]. Also, higher-dimensional systems can allow for the compression of qubit non-local gates into local gates on a single qudit, significantly reducing decoherence effects. It is possible to tailor the qudit architecture for the purpose of compressing the qubit circuit, hence compress the quantum circuit, and produce better circuit runtimes with lower error rates, as seen in [11], [54]–[57]. An example of this approach will be described in *Chapter* 8.

Application Tailored Architectures We have seen that custom mixed-dimensional architectures can enhance the quality of the runtime of qubit circuits. However, this is not the only solution that mixed-dimensional architectures enable, because a tailored architecture can be used for a native implementation

of the target applications. This alternative strategy that leverages efficient interactions between qubits and qudits of different dimensions produces less error-prone circuits. This is possible due to the direct match between the algorithms and the operations available on the device, which can be translated in more specific and optimized sequences of native gates. Eventually, this strategy could be the long term winner implementation for algorithms naturally requiring multi-level states.

Practical examples are the tailoring of the architecture to represent efficiently problems that inherently require qudit units of information [58], or the loading of classical data on a quantum computers, that will be more effectively manipulated for quantum machine learning approaches [59], [60]. The influence of this codesign approach of mixed-dimensional quantum systems reaches several computational fields: within optimization and search algorithms, these systems can lower the computational complexity and improve performance, especially in problems demanding efficient integer variable representations [61]–[63].

Simulating physical systems is another critical area of application that greatly benefits from mixed-dimensional quantum computers. These systems are particularly suitable for simulating high-spin states, which allows for deeper insight into complex physical interactions [64]–[66]. Mixed-dimensional quantum computers have significantly advanced the modeling and execution of real-time simulations [67], [68], providing significant support for studying fermion-boson interactions, quantum electrodynamics, and pushing discoveries in field theories [69]–[72].

Finally, in quantum chemistry, the tailored architectures facilitate more accurate modeling of molecular interactions and dynamics, expanding the possibilities for computational chemistry applications [73]. The investigation of lattice gauge theories has also benefited significantly from qudit-based approaches, providing new insights into fundamental physics and particle interactions [68], [74]–[76].

This flexibility of mixed-dimensional systems has resulted in remarkable advances in the reliability of quantum circuits and the efficiency of computations across various applications. However, it is important to acknowledge that simply increasing the dimensions of the qudit without adequate considerations does not fully leverage their potential [77], [78]. Designing mixed-dimensional systems requires a balanced approach that accounts for both the benefits and limitations of higher-dimensional spaces.

In conclusion, mixed-dimensional quantum systems offer a comprehensive range of architectural and algorithmic advantages that enhance the design and efficiency of quantum circuits. Their applications span from optimization and search algorithms to the simulation of complex physical systems, demonstrating remarkable versatility and potential. As universal qudit quantum processors continue to mature and become more sophisticated, their potential to revolutionize quantum computing across diverse fields will only grow, promising exciting developments in both theoretical understanding and practical applications.

2.3 Basics of Design Tools for Quantum Computing

Quantum computers are receiving increasing support from software tools and ecosystems, such as those related to *High-Performance Computing* (HPC). This growing support is accompanied by significant advances in qubit systems in terms of architectural designs [79]–[82], fault-tolerant quantum computations [83]–[86], and co-design strategies [87], [88]. Despite these developments, qubits represent only a fraction of the full computational potential of quantum computers. To move beyond this limitation, mixed-dimensional quantum computing offers a complementary approach by combining the capabilities of qubits and qudits. This paradigm promises better computational capabilities [9], [74] and better opportunities for hardware and application co-design.

However, current quantum software remains predominantly focused on and optimized for qubit-based implementations. The absence of software frameworks for mixed-dimensional systems, coupled with a lack of automated methods, significantly slows the progress of this emerging paradigm. Consequently, the development of new software tools and frameworks is essential for accelerating the deployment of mixed-dimensional quantum devices. Fortunately, some of the tasks quantum software needs to solve have already been identified, such as classical simulation (*Part* II) and compilation (*Part* III).

Designing quantum software presents challenges primarily due to two factors: the exponential memory growth associated with representing quantum states and operations and the extensive parameter space that needs to be explored. However, when it comes to mixed-dimensional quantum systems that integrate qubits with higher-dimensional qudits, these challenges intensify. This requires the development of more sophisticated data-structures and methods to manage these complexities. This section builds the background necessary for making this thesis self-contained and lays the theoretical foundation that quantum software needs for tackling the design challenges of mixed-dimensional quantum computing. It specifically illustrates decision diagrams (Section 2.3.1), tensor networks (Section 2.3.2), and other theoretical principles crucial to understanding the compilation process (Section 2.3.3, Section 2.3.4). These foundational elements set the stage for the quantum design methods discussed throughout this thesis.

2.3.1 Decision Diagrams Basics for Quantum Circuit Simulation

Since the 1980s, Boolean functions have been associated with traditional electronic circuits and systems often involve many inputs and outputs, rendering explicit forms like truth tables impractical due to their exponential expansion. As a result, design automation tools necessitate more compact representations that retain accurate functionality without loss of detail. A commonly employed method is the use of decision diagrams, which represent Boolean functions as directed acyclic graphs (DAGs) [89]. These graphs are constructed by recursively breaking down the function according to its input variables, achieving compactness through shared nodes that depict redundant sub-functions. Decision diagrams facilitated the emergence of various forms, such as BDDs, FBDDs, KFDDs, MTBDDs, and ZDDs (see, for instance, [90]–[96]). Building on their historical success, decision diagrams have been recommended for use in quantum computing [97]–[101], where truth tables are replaced by matrices. Moreover, recently, they have attracted considerable attention for design tasks such as simulation [102], [103], and synthesis [104]–[107] of quantum circuits.

This section introduces the basic intuition of decision diagrams for representing quantum states and operations that will be expanded by the advancement in decision diagrams for mixed-dimensional systems, in *Chapter* 4.

Quantum States as Decision Diagrams We have seen in Section 2.2.1 that the state of a quantum system can be represented by a vector of complex numbers which are normalized, so that their squared

modulus summed is equal to one. It is also useful to notice that the state is a linear combination of basis states, that we can pick to be the computational basis (or Z basis), and we can associate to each complex number a state in the basis, as in Eq. (2.24).

$$|\Psi\rangle = \begin{bmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2} & 0 & \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2} & 0 \\ |000\rangle & |001\rangle & |010\rangle & |011\rangle & |100\rangle & |101\rangle & |110\rangle & |111\rangle \end{bmatrix}^T \tag{2.24}$$

Since each element of the basis is the Kronecker product of the basis of each unit of information in the system, we can recursively decompose each basis into different levels where each level is a fan of the possible choices for an input variable representing a unit of information, and as we descend in the hierarchy, we split the space of the possible representations as in Eq. (2.25).

$$\overbrace{\overbrace{\overbrace{|000\rangle}^{|00q_0\rangle}\ \overbrace{|001\rangle}^{|01q_0\rangle}\ \overbrace{|010\rangle}^{|10q_0\rangle}\ \overbrace{|011\rangle}^{|11q_0\rangle}}^{|0q_1q_0\rangle}\ \overbrace{\overbrace{|100\rangle}^{|00q_0\rangle}\ \overbrace{|101\rangle}^{|01q_0\rangle}\ \overbrace{|110\rangle}^{|10q_0\rangle}\ \overbrace{|111\rangle}^{|11q_0\rangle}}^{|1q_1q_0\rangle}}^{|q_2q_1q_0\rangle} \tag{2.25}$$

We can represent the recursive split in a data-structure with a Decision Diagram (DD), a Directed Acyclic Graph (DAG) made of nodes and edges that resemble a tree. Each node has an incoming edge that is the entry point, and each node of the DD performs the split in terms of logic, where the outgoing edges, called *successor edges*, will point to the next nodes, being themselves an incoming edge. Usually, the left-most edge is the lowest logic value taken for the variable at that level, and the right-most is the highest logic value. The amplitudes are noted on the edges; if not noted, they are automatically assigned 1. There are two *terminal nodes*, the node *one* and *zero*, beyond which the DD does not exist and their corresponding incoming edges assume values one and zero. In Figure 2.2, the state in Eq. (2.24) is translated in a primitive form of a decision diagram, where we can easily reconstruct the logic of the state and how to reconduct each basis to the corresponding complex entry. To easily keep track of which node represents which qubit or qudit, the nodes are labelled accordingly.

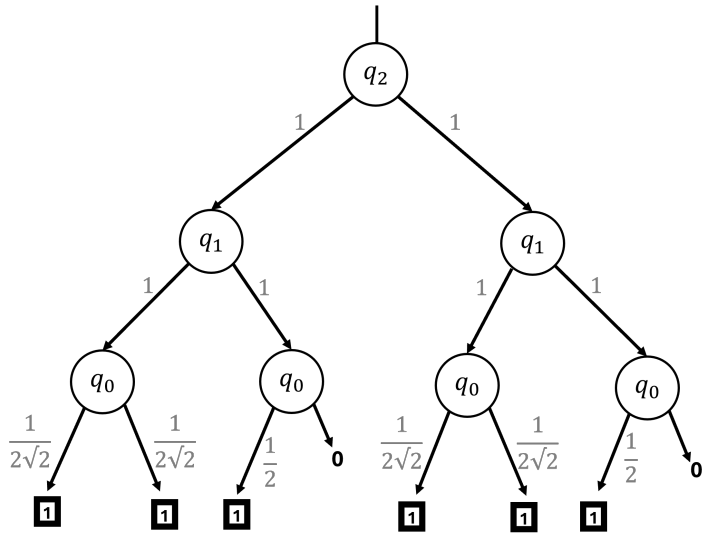


Figure 2.2 A decision diagram in the most primitive form, just nodes and edges.

Given a decision diagram, a complex entry of the state is retrieved by multiplying all the values on the edges, from the first node (the *root*) up to the terminal node, which is Figure 2.2 would be multiplying all the ones until finding the actual complex number at the leaves of the DD. It is an intuitive representation that at

the current stage we can define it as a primite DD, since there is no advantage in using this data-structure to represent a state, there are still several key features missing in this description.

Since in the decision diagram it is possible that several subtrees at the same level could be identical, and present in this way just redundancy in the representation, it is possible to store the subtree only once and merge the representations by redirecting the edges. This procedure is called *reduction* of the decision diagram. Figure 2.3 shows an example of a reduced DD.

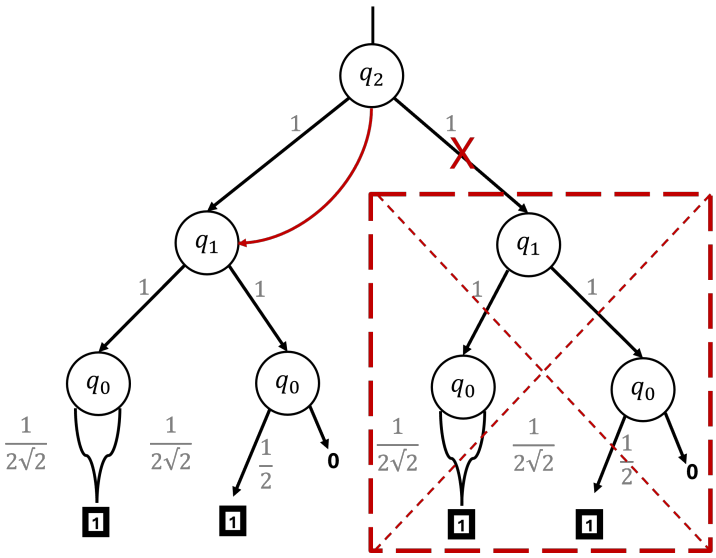


Figure 2.3 A decision diagram where the edges are redirected to reduce its redundancies.

Although, in this particular case, this change in the decision diagram led to almost 50% reduction, in most cases, the success of the procedure is bounded by an extra processing step. We are talking about the *normalization* of the nodes in the DD [100], [103]. This is a procedure that is applied recursively bottom-up and in simple terms, it divides the weight on the outgoing edges by a factor and adjusts the incoming edge of the node by that factor. The normalization of the state DDs is usually performed by dividing the weight of each edge by the norm of the list of weights of the outgoing edges, while the incoming edge is multiplied by that factor. The normalization ensures that the sum of the squared magnitude of the outgoing edges sums up to one, following the quantum information rationale and representing the possibility of measuring the logic state. Additionally, the normalization allows to reflect the numerical structures of the state and which information is stored in the leaves throughout the DD, making possible to identify redundancy early on from the root. Figure 2.4 shows an example of a reduced and normalized DD.

From the figure it is possible to easily verify that the sum of the squared magnitudes of the outgoing edges of each node sum up to 1. In fact, $(\frac{1}{\sqrt{2}})^2 = 0.5$ and, if summed, $0.5 + 0.5 = 1$, valid for nodes q_2, q_1 and q_0. The normalization also allows the propagation of knowledge about the presence of several zeros in the state. In fact, the normalization would propagate the 0 factors toward the root up to a point where then the whole subtree would be cut from the structure, and the outgoing edge would directly point to the terminal zero. To this end, the early termination allows DD to efficiently represent sparse states, like in Figure 2.5.

It is fundamental to explain that the decision diagram of the state is not constructed by following the order of the explanation made so far, which was just for the sake of intuition, but it is always constructed in the minimal compact form where possible. *Chapter* 4 will clarify the construction process. Moreover, the number of nodes in the decision diagrams corresponds to its *size* and it will be the measure of compactness and a metric of complexity for algorithms exploiting this structure.

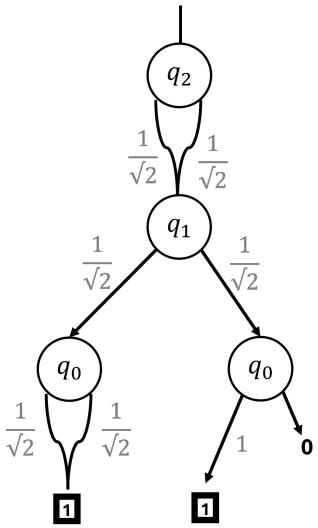


Figure 2.4 A reduced and normalized decision diagram.

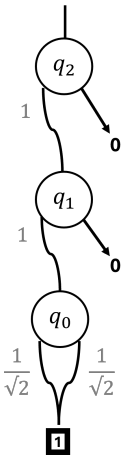


Figure 2.5 Decision diagram with early termination.

Quantum Operations as Decision Diagrams As we have observed in Section 2.2.2, quantum operations are represented by unitary matrices made of complex numbers. In Figure 2.6 it is possible to see that the construction of a matrix DD is just an extension of the state vector case, where instead of modeling possible measurement outcomes for each qubit or qudit, the outgoing edges model the multilinear map represented by the matrix.

The first outgoing edge represents the function where the lowest logic state is mapped with a linear function (a complex number) again to the same state, the second edge represents how the state $|0\rangle$ is linearly mapped to the state $|1\rangle$, and so on, as depicted in Eq. (2.26). The construction similar to the state is done recursively by dividing the matrix in $d \times d$ block at each time. The reduction and sparsity are again accounted for, thanks to a normalization scheme of the matrix DD, that this time is dividing each weight by the largest magnitude among the outgoing edges, and multiplying the incoming edge by that factor. This last procedure is implementation-dependent and could change depending on the goal for which DDs are

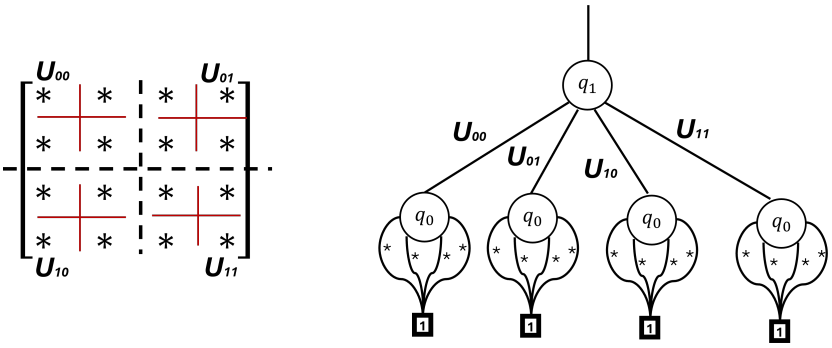


Figure 2.6 Translation of a matrix into a DD.

involved. Here, we show how the tensor product between a qutrit and a qubit is converted into a decision diagram. One is the X operation on the qutrit, and the other is an i phase on a qubit. [1]

$$X_3 \otimes iI_2 = \begin{bmatrix} 0 & 0 & \vdots & 0 & 0 & \vdots & i & 0 \\ 0 & 0 & \vdots & 0 & 0 & \vdots & 0 & i \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ i & 0 & \vdots & 0 & 0 & \vdots & 0 & 0 \\ 0 & i & \vdots & 0 & 0 & \vdots & 0 & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \vdots & i & 0 & \vdots & 0 & 0 \\ 0 & 0 & \vdots & 0 & i & \vdots & 0 & 0 \end{bmatrix} \tag{2.26}$$

The tensor product here acts as if it embedded iI matrices, and zero matrices in the entries of the qutrit matrix.

$$\begin{bmatrix} \underline{0} & \underline{0} & iI \\ iI & \underline{0} & \underline{0} \\ \underline{0} & iI & \underline{0} \end{bmatrix} \tag{2.27}$$

In Figure 2.7, the nodes on the bottom show the identity matrix, connected on top to the X matrix. The DD is reduced, sparse and normalized.

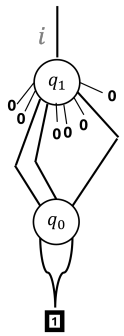


Figure 2.7 Translation of the $X_3 \otimes iI_2$ matrix into a DD.

[1] The global phase in the matrix is physically neglected, but can still be represented numerically. The example is artificial for the sake of clarity.

Operations on DDs After presenting states and matrices as DDs, the last element to explain in quantum computations, as DD, is how to perform the operations, which reduces to being able to perform matrix-vector multiplications between the DD forms. It is logical to imagine that the operations will extend concepts coming from matrix-vector multiplication, and given the tree-like structure, it will use recursively divided operations in subcomputations, matching the decomposition of the DD. As the operations follow the data structure, the complexity of the operations will be inherently influenced by the size of the operation and state DDs. This topic will be explained in detail in *Chapter* 4.

2.3.2 Tensor Networks Basics for Quantum Circuit Simulation

Tensor networks, such as Matrix Product States (MPS), Projected Entangled-Pairs States (PEPS), and the Multiscale Entanglement Renormalization Ansatz (MERA), are especially suited for the study of quantum many-body problems. The selection of the appropriate method depends on the model being analyzed, [108]–[111] and these networks have become crucial in the realm of quantum many-body physics for accurately simulating systems in one, two, and three dimensions [112], [113] and examining their characteristics. Different in their specific implementations, these networks essentially comprise a collection of tensors interconnected via shared indices. This setup can be graphically represented by illustrating tensors as nodes within an undirected graph, with edges denoting the common indices among the tensors, like in Figure 2.8.

Each tensor is a multi-dimensional array comprising complex values, where a tensor's rank specifies the number of dimensions (or indices) it possesses, whereas its shape specifies the number of elements along each dimension. Conforming to algebraic principles and expressed using Einstein's summation convention, tensors are subject to a contraction process, explained in Example 9. The efficiency of tensor network contractions is influenced by the dimensions and configuration of the tensors, rather than their specific contents. This enables effective prior estimation of contraction plan performance, yet highlights that tensor networks require a strategically chosen contraction plan to be beneficial. In order to derive valuable insights from a tensor network, we usually need to sequentially contract its tensors in pairs until only a single tensor remains.

Example 9. *Consider three square matrices A, B, and C, each belonging to $\mathbb{C}^{n\times n}$. The multiplication $C = AB$ can be detailed as $C_{ij} = \sum_{k=0}^{n-1} A_{ik} \cdot B_{kj}$, where indices i and j range from 0 to $n-1$. This process involves contracting [114] the rank-2 tensors $A[i,k]$ and $B[k,j]$ using their common index k. Practically, we multiply elements from matrix A with those from matrix B. Specifically, we select an element from the i_{th} row of A and a corresponding one from the j_{th} column of B, maintaining fixed rows and columns respectively. The index k in both A_{ik} and B_{kj} identifies which elements to multiply, aligning those with shared k values. Next, the addition occurs: After performing all multiplications, we sum all the resultant products. We execute this for each k value, beginning at $k = 0$ and continuing until $k = n - 1$, with n as the dimension size. This total provides the value sited at (i, j) in the resultant matrix C. In summary, we pair and multiply each corresponding element, then aggregate the products to obtain each component in the final matrix. Figure 2.8 show graphically how to exploit the notation.*

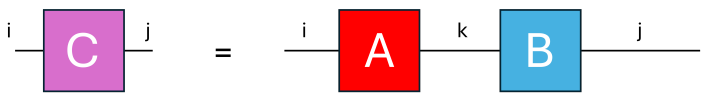


Figure 2.8 Example of contraction of two bidimensional tensors.

Quantum states and operations as tensors Tensor networks, similar to decision diagrams, efficiently represent a quantum system's starting state and processes. In the case of quantum circuits simulation, traditionally MPS are employed, where the initial and final states are depicted as collections of rank-1 tensors. The quantum operations are applied to the corresponding rank-1 tensors, by connecting in the network n-qudit gates, illustrated by a rank-$2n$ tensor and linked to adjacent tensors via common indices.

The primary difficulty is deciding upon a sequence of contractions that minimizes the size of virtual indices χ_i (bond dimension) and the configuration of intermediate tensors, since these aspects are crucial for computational efficiency. When it is possible to keep both the bond dimension χ_i, a virtual index, and the size of intermediate tensors at a reasonable level, tensor networks facilitate efficient computations. This sequence is known as the contraction plan. Determining the most efficient contraction sequence is known to be an NP-hard problem [115], which has consequently become a major area of research in recent times [116]–[119].

Figure 2.9 illustrates a quantum circuit transformed into Matrix Product State (MPS) form, a specific category of tensor network. The tensors are interconnected through common indices. The figure depicts the contents of the tensors, although these do not affect the selection of the contraction plan. While the contents may vary, what is crucial in determining the contraction paths is the size of the indices in the network.

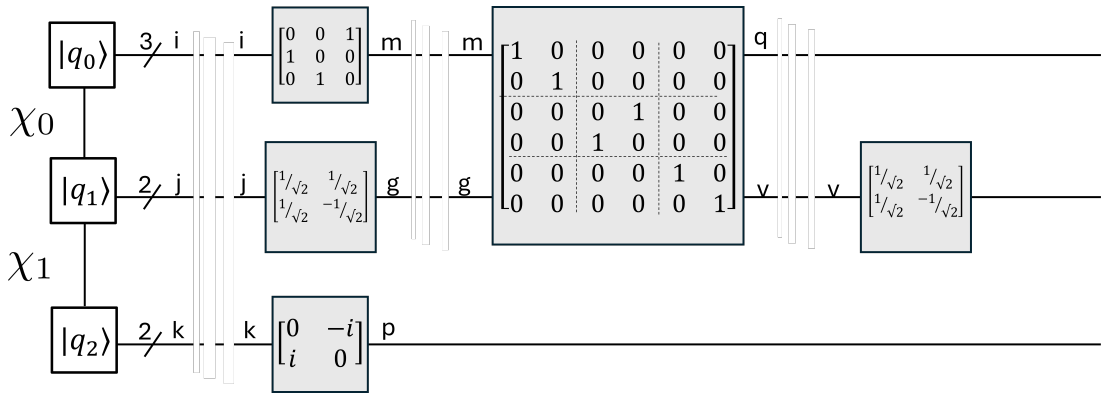


Figure 2.9 A MPS tensor network representing a quantum circuit.

The sections on Decision Diagrams (DD) and Tensor Networks (TN) covered so far are fundamental for the simulation of quantum circuits and will be essential for enabling the simulators developed in *Part* IV. We proceed now by covering the most useful concepts that will facilitate the comprehension of *Part* III.

2.3.3 Quantum Information Guidelines for Quantum Circuits Compilation

Quantum compilation represents a fundamental challenge in quantum computing: the translation of abstract quantum algorithms into executable sequences of physical operations. At its core, there is a reconstruction problem where we seek to build quantum circuits that faithfully implement desired quantum transformations while minimizing errors and resource requirements. The complexity of this task stems from the need to preserve crucial quantum properties like entanglement and superposition throughout the compilation process. While traditional approaches treat this as a decomposition problem, which is breaking down arbitrary unitary operations into sequences of elementary gates, modern quantum compilation incorporates optimization techniques as well. This includes using heuristic searches to find more efficient implementations that minimize circuit depth, gate count, and error accumulation while respecting the hardware's physical constraints.

In quantum compilation, selecting appropriate metrics [120] is crucial for optimizing and validating the quantum circuits produced. It is useful to have several metrics since each captures different aspects of quantum information processing, and, in fact, no single benchmark can completely characterize a system. Given the nature of quantum information, the common element to all metrics are measurements, which are at the base of the evaluation of how well the compilation was performed. Given the vastity of the topic we refer to [120].

It is possible to draw the probability of measuring a specific basis state from a set of measurements. If one wants to assess how well a circuit can reconstruct a specific state, we would have to compare the distribution of measurements obtained from the circuit, with the distribution sampled from the target state. At the classical level, when examining quantum measurement probability distributions, we begin with basic metrics such as the total variation distance [121]

$$\delta(p, q) = \frac{1}{2} \sum_i |p_i - q_i| \tag{2.28}$$

which intuitively measures distributional dissimilarity by highlighting the largest difference in the probability of any event occurring according to these distributions. Additionally, information-theoretic metrics like the relative entropy [122]

$$D(p||q) = \sum_i p_i \log(p_i/q_i) \tag{2.29}$$

provide deeper insights into the statistical differences between distributions, serving as a type of statistical distance to quantify the discrepancy between a model probability distribution Q and a true probability distribution P.

When we move to quantum states, the metrics become richer to capture quantum-specific phenomena like superposition and entanglement. The trace distance [1], [123]

$$D(\rho, \sigma) = \frac{1}{2}\mathsf{Tr}|\rho - \sigma| \tag{2.30}$$

that represents the maximum probability of distinguishing two quantum states through any measurement, making it operationally meaningful.

Since two states represent potentially two distributions, we could use quantum state fidelity to measure how much they overlap [1], [123]

$$F(\rho, \sigma) = \mathsf{Tr}\sqrt{\sqrt{\rho}\sigma\sqrt{\rho}}$$

which could be interpreted as the probability of one state passing a test for being the other state. These metrics are related through important inequalities: $1 - F \leq D \leq \sqrt{1 - F^2}$, with D being the trace distance.

The metrics presented so far can be useful for the problem of state compilation, but for complete quantum processes, which are the primary concern in compilation, we need metrics that capture not just state similarities but the entire quantum channel behavior. In that case, we could use the diamond distance [124]

$$|\mathcal{E} - \mathcal{F}|_\diamond = \frac{1}{2}|\mathcal{I} \otimes (\mathcal{E} - \mathcal{F})|_1 \tag{2.31}$$

that generalizes the trace distance to quantum channels, measuring the worst-case distinguishability between processes while accounting for potential entanglement with auxiliary systems.

Another way to write the process fidelity [125] would become:

$$F_{\text{process}}(\mathcal{E}, \mathcal{F}) = \frac{\mathsf{Tr}(J_\mathcal{E} J_\mathcal{F})}{\sqrt{\mathsf{Tr}(J_\mathcal{E}^2)\mathsf{Tr}(J_\mathcal{F}^2)}} \tag{2.32}$$

it extends state fidelity to processes through the Jamiołkowski representation $J_{\mathcal{E}}$ [126], [127], which maps quantum channels to quantum states in a larger Hilbert space. This representation is particularly powerful, as it allows us to treat process metrics as state metrics in an extended space, providing a unified framework for optimization. The relationship between process metrics parallels that of state metrics, but with additional complexity due to the channel structure. The diamond distance and process fidelity are linked by bounds similar to those for states, yet each reflects distinct aspects of process similarity.

In practice, these metrics play distinct roles in compilation: the diamond distance often serves as a rigorous bound for compiler correctness, while process fidelity often guides optimization algorithms. The choice between them depends on the compilation goal, whether we are more concerned with worst-case guarantees (favoring diamond distance) or average performance (favoring fidelity measures).

While all these concepts are valuable, they are often computationally challenging. Consequently, many compilers opt for a simpler metric, the Frobenius norm, which acts as a link across different fidelity and distance measures in quantum information theory. To compute it, one squares each matrix element, sums all these squares, and then takes the square root. Formally, for a matrix A, the Frobenius norm is expressed as:

$$\|A\|_F = \sqrt{\sum_i \sum_j |a_{ij}|^2} \tag{2.33}$$

Despite its lack of physical interpretability, it is especially valuable for providing bounds—both upper and lower—on more physically intuitive measures such as trace distance and fidelity. Although these metrics offer a broader understanding of the objectives of quantum compilation, there are always other valid alternatives.

Respecting fidelity metrics will provide help to automated methods in evaluating the quality of the results; however, there are properties of states and quantum properties like entanglement that must be reproduced within the quantum circuit. As a result, achieving a balance between high fidelity and the preservation of these essential quantum properties is crucial for the success of quantum computation. In *Part* III, fidelities will be extensively used and explained in the context of their role in quantum compilation.

2.3.4 Heuristic Search Methods for Quantum Compilation

In the previous section, we have outlined quality metrics to evaluate the compilation process. These metrics can be used to guide automated solvers in optimizing the translation of abstract quantum algorithms into executable sequences of physical operations, or for the optimization of sequences found, where the goal could be to minimize the number of operations, reducing noise, or decreasing the number of qudits in the circuit. Due to the vast space of potential quantum circuits for a single algorithm, finding the global optimal solution is often unfeasible. In such instances, automated procedures can utilize heuristic search-based algorithms.

Heuristic search methods are the cornerstone of problem-solving techniques in computer science. These are educated guesses or rules of thumb, that guide the search process, often allowing for efficient solutions to complex problems. This section provides an overview of the foundations of heuristic search, and the challenges faced when applying these techniques, particularly in the context of quantum compilation.

A heuristic function [128], [129], denoted $h(n)$, provides an estimate of the cost to reach a goal state from a given state n. Its utility in guiding the search process depends critically on the quality and properties of the heuristic function, and in our case it will be used to guide the compilation process to reach an efficient implementation of a quantum circuit.

Key Properties of Heuristic Functions Effective heuristic functions in quantum computing exhibit three fundamental properties that work together to ensure both theoretical soundness and practical utility. The first

essential property is *admissibility*, which requires that the heuristic function $h(n)$ never overestimates the actual cost to reach the goal and guarantees that our algorithms will find optimal solutions when they exist. Based on admissibility, the second key property is *monotonicity*, where for any state n and its successor n' separated by a step cost c, the function must satisfy $h(n) \leq c + h(n')$. This relationship ensures that the paths to our goal state remain well-ordered throughout the search process, preventing algorithmic confusion and maintaining search efficiency. The third property, *informedness*, speaks to the practical utility of our heuristic function. This property captures how accurately $h(n)$ approximates the true cost to reach the goal. The degree of informedness directly influences the computational efficiency of our search process, as a more informed heuristic can dramatically reduce the number of states we need to explore, while a less informed one might offer little advantage over uninformed search methods. These properties work together to create heuristic functions that are both theoretically sound and practically useful in quantum computing compilation, with the potential optimality of the solutions and the efficiency of the search process.

Example of Heuristic Search Algorithms in Quantum Domains

Heuristic search methods find particularly fertile ground in quantum computing applications, where they must navigate vast yet highly structured search spaces. One way of conceptualizing these spaces is as graphs. Each node in these graphs represents a partial solution, while the edges between these nodes correspond to quantum gates or state transitions in circuit form. Complete paths through the graph represent sequences of quantum operations that transform an initial state toward a desired goal state—whether that is achieving a specific unitary transformation or minimizing a carefully chosen objective function.

Best-First Search algorithms have proven especially valuable in exploring these wide spaces, although they require careful adaptation to handle the unique challenges of quantum systems. These algorithms make their traversal decisions by prioritizing nodes based on an evaluation function, always selecting the most promising state for exploration. However, the high dimensionality of circuit design spaces demand sophisticated modifications to traditional backtracking approaches.

A Search* algorithm exemplifies the adaptation to quantum contexts. It employs an evaluation function $f(n) = g(n) + h(n)$ that bridges past and future alternatives. The component $g(n)$ represents concrete costs already incurred, such as the number of quantum gates applied in the circuit, while $h(n)$ looks forward, estimating the remaining distance to the goal, which could be measured as the fidelity gap between the current and the target quantum states, or how many gates are still to be applied. This balanced approach helps navigate the complex landscape of quantum circuit compilation efficiently while maintaining the quality of the solution. However, it works only for specific classes of problems. Figure 2.10 shows graphical examples of heuristic searches. *Chapter* 11 proves the usefulness of heurstic searches.

Simulated Annealing and Local Search Algorithms

The application of search algorithms to quantum computing domains presents both unique challenges and promising opportunities. Although these methods have proven successful in classical optimization, their adaptation to quantum settings requires careful consideration of the underlying physics and computational constraints. The iterative evolution of candidate solutions in variational quantum circuits, for instance, must contend with the inherent noise and decoherence effects present in current quantum hardware. Simulated Annealing (SA) represents a particularly intriguing direction at the intersection of classical optimization techniques and quantum computing. Its ability to navigate rugged energy landscapes offers potential advantages over classical approaches, especially when dealing with the complex parameter spaces typical in quantum circuit optimization. Future research might also explore how quantum-specific noise models

could be incorporated into heuristic functions, potentially turning what is typically viewed as a limitation into a feature that guides the search process more effectively.

Chapter 10 will describe how simulated annealing and local searches help the compilation of quantum gates.

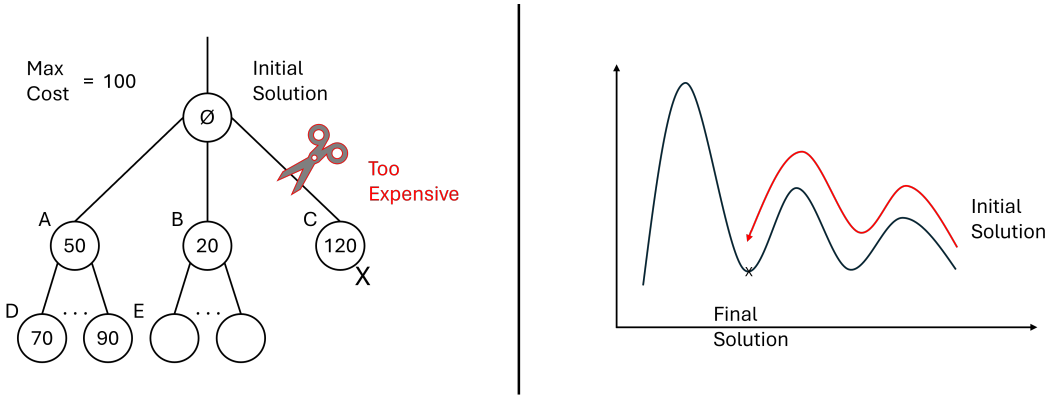


Figure 2.10 Two simple visual examples of two heuristic search methods: on the left, a graph-based search method, on the right, a heuristic optimization function that searches a continuous landscape looking for the optimum.

Designing Heuristics for Quantum Compilers

The design of effective heuristics for quantum software builds upon the classical foundation of heuristic functions that guide search processes. Quantum heuristic design must adapt and extend these concepts with the use of quantum information. When developing heuristics for quantum compilation, several key principles emerge as essential for ensuring their effectiveness. *Accuracy* stands as a major concern as the heuristic must provide reliable estimates of quantum-specific distances, whether measured through fidelity metrics or trace distances. The *efficiency* of heuristic computation remains crucial, particularly given the rapidly growing dimensionality of quantum systems. As the size and complexity of quantum problems scale up, the heuristic must maintain computational tractability while still providing meaningful guidance. This balance between accuracy and computational efficiency becomes especially delicate, where the state space grows exponentially with the number of qubits. The principle of *monotonicity* represents a natural evolution of classical consistency requirements adapted for the quantum domain. This demands careful consideration of operator norms and other quantum-specific metrics that capture the unique behavior of quantum systems. The heuristic must not only provide lower bounds on costs, as in classical cases, but must also respect the fundamental constraints imposed by quantum mechanics. These principles work together to create heuristics that can effectively guide quantum compilation processes while maintaining physical validity and computational efficiency. They provide a framework for developing practical tools that will be explored in *Part* III.

Challenges and Limitations The challenges of designing quantum heuristics extend far beyond the basic principles. At the forefront of these challenges lies the complexity of search space. The exponential growth of the search space with each additional qubit/qudit creates a fundamental scaling challenge that classical approaches simply cannot handle directly. This necessitates the development of heuristics capable of operating within reduced or approximate representations of the quantum state space, finding clever ways to capture essential quantum properties while maintaining computational tractability. This stochastic behavior means that heuristics must be designed to handle uncertainty and often require multiple sampling iterations to build reliable estimates of system properties when needed.

Classical heuristics, originally developed for deterministic and separable systems, require significant modification to effectively address entangled states or superpositions. Additionally, these heuristics must accommodate practical hardware constraints such as limited coherence times of quantum states, imperfect gate fidelities, and restricted qubit counts in current quantum devices. This may involve integrating error models directly into the heuristic or devising noise-resilient approaches that remain effective despite environmental disturbances. These strategies must balance the competing demands of accuracy and practicality, ensuring that heuristics remain efficient in real-world quantum systems. However, many considerations require knowledge of quantum dynamics, which is often challenging to compute efficiently due to the inherent complexities of quantum problems.

The background we just built on the design of heuristic functions will be fundamental for understanding the choice made to solve compilation problems in *Part* III, especially in *Chapter* 10 and *Chapter* 11.

Part II

Classical Simulation of Quantum Circuits

3 Overview of Part II

The advent of mixed-dimensional quantum architectures, which integrate qubits and higher-dimensional qudits, urgently necessitates reliable simulation tools. Despite the availability of several well-developed simulators for qubit systems, there is a significant lack of comprehensive frameworks for simulating mixed-dimensional quantum computing. This shortfall considerably hinders the quantum computing community's efforts to evaluate the potential benefits of these architectures.

Classical simulation serves several critical roles, such as retrieving otherwise inaccessible information about quantum states and enabling the exploration and verification of quantum applications when appropriate hardware is either unavailable or limited in scale or precision. For mixed-dimensional systems, these capabilities are particularly valuable for debugging and validating quantum programs and hardware, as well as establishing baselines to evaluate various mixed-dimensional architectures. This is especially relevant for mixed-dimensional quantum systems, where the interaction between qubits and qudits introduces additional complexity to both design and validation. While the ultimate goal is to execute computations on actual quantum hardware, classical simulation remains indispensable, particularly during the design and development stages.

Simulating quantum computations consists of transforming an initial quantum state via a series of operations, outlined as a quantum circuit, before measuring the resultant state. Despite being conceptually straightforward, similarly to matrix-vector multiplication, the simulation procedure is severely limited by memory demands, especially in mixed-dimensional systems where higher-dimensional qudits aggravate state-space complexity. As a result, naive state vector techniques become quickly unmanageable, even with an abundance of computational resources.

Part II of this Ph.D. thesis examines the application of data structures, namely decision diagrams, to address simulation challenges and manage the memory demands of simulations. It looks into ways to modify and refine these structures for mixed-dimensional quantum systems and considers integrating mixed-dimensional noise into classical simulations, highlighting potential advantages for decision diagrams. The main contributions of our work on simulation are presented in two chapters. *Chapter* 4, based on [10], introduces a novel proposal in which we developed a specific kind of decision diagram designed to manage the simulation of mixed-dimensional quantum circuits, along with its related implementation. *Chapter* 5 describes a first formalization of noise models for mixed-dimensional quantum computers, which is more accurately aligned with experimental results and may offer greater advantages compared to the qudit depolarizing noise channel in simulations.

4 Mixed-Dimensional Quantum Circuit Simulation with Decision Diagrams

Advanced data-structures, such as decision diagrams (Section 2.3.1) and tensor networks (Section 2.3.2), have shown potential to mitigate the substantial complexity involved in classical quantum circuit simulation for various practically relevant scenarios. Considering the highly optimized performance of decision diagrams for qubit-based quantum computers, extending their use to new, unexplored paradigms could be advantageous for simulating new architectural types. In fact, the increased flexibility of decision diagrams could significantly benefit mixed-dimensional systems.

In this chapter, based on [10], we contribute to that by introducing the basis for a new classical simulator for mixed-dimensional quantum circuits, i.e., circuits where each qudit may have a different dimensionality. To this end, we propose an extension to edge-weighted Decision Diagrams [98], [101], [102] which serve as the main data-structure to cope with the (exponential) complexity of simulation. In the past, decision diagrams have been proven to be a suitable data structure to compactly represent exponentially-sized data in many cases [98]–[101], [105], [130]. The type of decision diagram proposed in this work dynamically captures the dimensionality of each qudit in the mixed-dimensional system to minimize the required memory. Further, it elegantly visualizes the individual dimensionalities of the qudits.

This chapter is structured as follows. Section 4.1 motivates the problem of quantum circuit simulation, details the contributions of this paper, and gives an overview of the state-of-the-art in quantum circuit simulation. Section 4.2 describes the type of decision diagram proposed for a mixed-dimensional system and Section 4.3 details the corresponding implementation. Finally, Section 4.4 summarizes the experimental evaluation.

4.1 Motivation

In this section, we investigate the problem of efficiently simulating quantum circuits where the qudits can have different dimensions, referred to as mixed-dimensional systems. We review as well the design automation task of quantum simulation that is going to enable the utilization of mixed-dimensional systems. Following that, we discuss the challenges of simulating quantum computations and the benefits of simulating quantum circuits of mixed dimensionality.

4.1.1 The Need for Quantum Circuit Simulation

The emergence of practical examples of mixed-dimensional quantum architectures, integrating both qubits and higher-dimensional qudits, highlights the urgent requirement for reliable simulation tools. For these systems, there is a significant lack of comprehensive frameworks for simulating mixed-dimensional quantum circuits, in addition to the critical demand for novel automated methods, software frameworks, and theoretical advancements, as evidenced by various successful examples [11], [13], [14], [131], [132]. This unaddressed deficiency considerably restrains the quantum computing community's capacity to evaluate the potential benefits of such architectures. Classical simulation is essential not only for understanding quantum

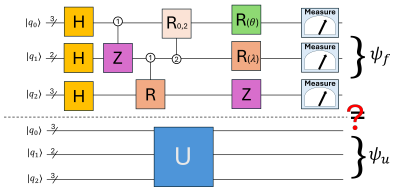


Figure 4.1 Verification of a quantum circuit given the unitary representing original algorithm.

computations but also for facilitating the exploration of new applications and hardware capabilities introduced by mixed-dimensional systems.

There are four immediate advantages to having an appropriate simulator for mixed-dimensional quantum systems.

1. **Getting otherwise opaque information about the quantum state**
   The inner processes of a quantum computer are fundamentally opaque. Observing a quantum state will inadvertently make it decay into a basis state – immediately destroying any superposition or entanglement. However, we can accurately calculate the exact information of a quantum state by classical means. Of course, this is only possible for smaller systems due to the exponential overhead in the number of qudits. Nevertheless, classical simulation is an important tool for algorithm development and debugging. This becomes even more important when targeting higher- and mixed-dimensional systems. Their increased complexity also affects the developers of these systems, making automated software support a necessity to reduce errors and speed up the design.

2. **Enabling design exploration**
   Simulation can take on the role of a tool for design exploration. In fact, in qudit systems, it is overall possible to explore new trade-offs between using higher-levels in the single qudit and using ancillas. The first one is similar to the effect of using ancilla qubits, but with a simpler circuit complexity and hardware design [53]. On mixed-dimensional systems this can be achieved with a temporal expansion of the Hilbert space. This leads to smaller circuits that have a higher chance of succeeding due to the smaller noise accumulated. However, just increasing the dimension of all qudits in a quantum circuit still leaves room for improvement [77], [78]. A simulator enables the study of different circuits, composed of more or less ancilla lines, depending on the limitations and potential of the implementing platform. This can lead to a more accurate study of suitable circuits for dedicated platforms and applications.

   As seen in Section 2.2.4, several applications can now benefit from the use of mixed-dimensional architectures. Exploring implementations with different mixtures of dimensionality can be of aid in assessing what architecture is more naturally suited for which algorithm.

3. **Aiding in verification and validation**
   Classical simulation of quantum circuits also aids in verification and validation schemes. On the one hand, it helps circuit equivalence checking by simulating two circuits with identical initial states and comparing the output. This is an easier task than constructing the matrices and comparing these. Such schemes that incorporate quantum circuit simulation have been proposed [133], [134]. On top of that, it allows to compare the behaviour of a quantum circuit in experiment with the ideal execution, allowing the scientists to predict the potential quality of results, and validating if the quantum hardware is performing the tasks it is required to do [120]. Moreover, these comparisons with the simulation results make it is possible to extract information about the sources of noise of a quantum system and formalize their behaviour.

Example 10. *The use of a simulator could be of use for verifying that a compiled quantum circuit, a new decomposition, or a particular encoding preserves the intended original functionality defined in the algorithm. In Figure 4.1, this is illustrated using a simple use case, where a unitary has been compiled to a mixed-dimensional system and the simulation can tell us if, given an initial input state, the final states of the two are identical. If true, the circuit performs correctly the intended algorithm.*

4. **Identifying potential for compression**
   Finally, it is possible to see through simulations that the computation may happen only on specific levels of a qudit. This could allow the compression of a single qudit's computational space from a higher dimension to a lower dimension restricted to only the useful levels [135]. By this, the representations become more compact, reflecting into a competitive error rate and experimental control, typical of the state of art of qudit systems [9]. The simulation of mixed-dimensional systems opens new possibilities in the field of circuit compression. Two examples illustrate the ideas.

   Example 11. *Consider the following operation U:*

$$U = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & e^{\frac{2\pi i}{3}} & e^{\frac{-2\pi i}{3}} & 0 & 0 \\ 0 & e^{\frac{-2\pi}{i}3} & e^{\frac{2\pi i}{3}} & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.1}$$

   It is possible to note that the local operation is operating only on two levels of the original five-level system (i.e., of a ququint). Taken into account that many operations are applied to only a subset of the available levels, we can consider the original system to be of only two dimensions, and consequently remap the logic levels to $|0\rangle$ and $|1\rangle$. This results in a more compact operation of dimension 2×2, namely

$$U' = \begin{bmatrix} e^{\frac{2\pi i}{3}} & e^{\frac{-2\pi i}{3}} \\ e^{\frac{-2\pi i}{3}} & e^{\frac{2\pi i}{3}} \end{bmatrix}. \tag{4.2}$$

These are some of the advantages of classical simulation for mixed-dimensional circuits and are likely among the most significant. Although we have just seen the advantages of simulation, there is still a problem.

4.1.2 The Scaling Problem

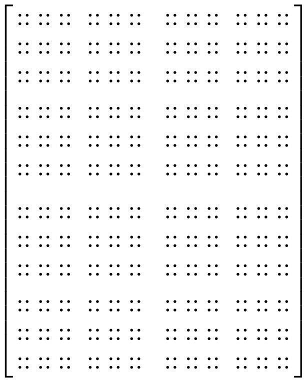
The simulation of a quantum circuit is conducted by successively applying operations in sequence to an initial quantum state, i.e., multiplying matrices and vectors. While quantum circuit simulation is simple, the exponential memory requirements in the number of qubits make it hard in practice. With qudits of higher dimensions, this memory requirement becomes even higher.

More precisely, given a system with N qudits of possibly mixed dimensions, a vector describing this system will have length $D = \prod_{i=0}^{N-1} d_i$, with d_i denoting the dimension of each qudit. Correspondingly, the matrices representing the operations on this system will have dimension $D \times D$. In fact, even for operations that affect only a subset of qudits, they have to be "padded", or "broadcasted", with the appropriate identity operations to ensure the correct (large) size (using the Kronecker product [1]).

Example 12. *Consider the Hadamard gate on a qutrit as shown in the background in Figure 2.1, also denoted H_3, affecting the second of the three qudits. Applying this operation leaves the first and third qudit*

untouched, i.e., it applies the identity operation of appropriate dimension. This is achieved by combining the Hadamard operation and identity operations via the Kronecker product as in the following illustration:

$$I_2 \otimes H_3 \otimes I_4 =$$



Here, each dot represents an element of the combined operation. The result is a matrix with 24×24 *entries.*

Here, Example 12 should convey an intuition about the rapid increase in size for state vector and operation matrices with respect to the number of qudits and their respective dimensionality. Consequently, the simulation of quantum computations requires an asymptotically exponential amount of memory with respect to the number of qudits. Single measurements of a quantum state suffer from the same complexity, but repeated measurements can be done in amortized linear time [103]. The memory complexity can be lowered quite significantly for many cases, especially for common single-qudit operations. In essence, the operation can be applied to the state vector directly, e.g., by swapping rows for the exchange-operation. Further, quantum states with many zero entries may be represented by sparse vectors [136]. However, the efficacy of this approach diminishes with operations affecting an increased number of qudits and quantum states that are not sparse.

4.1.3 State of the Art

There has been an ongoing and significant interest from researchers and engineers in developing solutions for simulating quantum computations. There are several available solutions that individually make use of different data structures, e.g., arrays [3], [137]–[139], decision diagrams [102], [130], tensor networks [140], [141], and matrix-product states [141]. However, the vast majority of these simulators have been developed so far with the focus on qubit systems and their related circuits, while qudit systems have not been provided with the same software support. Qubit simulators can be divided in two main categories.

The first category of simulators is referred to as *array-based*. These simulators have limitations due to their reliance on the straightforward representation of quantum states and operations. Typically, simple 1-dimensional and 2-dimensional arrays are used. To simulate larger quantum systems, they require massive hardware power, such as HPC infrastructures composed of thousands of nodes, and petabytes of distributed memory. The utilization of accelerators and dedicated hardware can improve the run-time of the matrix-vector multiplications.

The second category of simulators relies on specialized data structures. For example, solutions based on decision diagrams [102], [130], have been proposed to exploit redundancies to gain a more compact representation of state vectors and matrices. Simulators based on tensor networks, such as those found in popular software packages [142], [143], share a similar goal of simulating quantum systems. However, they achieve this goal by compressing the information using a network of tensors that are contracted together in a specific way. One example of such a simulation method is the *Matrix Product States* (MPS), which efficiently calculates physical properties of one-dimensional quantum systems [141]. There are also examples of simulators using a combination of matrix-vector multiplication and tensor networks as qsim [140], which supports a variety of circuit models.

However, the available implementations offer preliminary support of higher-dimensional systems at best, if at all. There are early examples of trials to qudit simulations of quantum circuits of homogeneous dimensions, that peaked with a proof-of-concept prototype of QMDDs [98]. More recent attempts come from the established framework for quantum circuits manipulation and simulation, Google's Cirq [140]. It provides the core functionalities for the simulation of quantum circuits but requires the scientists to explicitly integrate the software with the desired gate set and special circuit constructions. Hence, unfortunately, Cirq supports mixed-dimensional quantum circuit simulation only in principle. The qudit simulator QuDiet [144] provides a qubit-qudit quantum simulator package with quantum circuit templates of well-known qubit quantum algorithms for fast prototyping and simulation. QuDiet shares several similarities with Cirq and qsim [140] in terms of software structure and philosophy. So far, the existing simulators are focused on providing interfaces for the development of specific use cases rather than being ready-to-use qudit simulators. This lack of availability of the current software platforms and subsequent lack of automated methods is a major obstacle in development for higher- and mixed-dimensional systems.

4.1.4 Contribution

Motivated from the above, we present a mixed-dimensional quantum circuit simulator, based on decision diagrams, with a publicly available implementation. The aim is to investigate a new method for an efficient simulation of circuits with qudits of arbitrary and mixed dimensionality.

The proposed approach is motivated by the success of the decomposition schemes used in [98], [102], that fall under the family of *Decision Diagrams* (DDs; [98]–[101], [105], [130]) used in various fields of design automation, especially in simulation, verification, and synthesis. Further, the structure of directed acyclic graphs, which resemble tree, can elegantly illustrate the dimensionalities, interactions, and entanglement in mixed-dimensional systems.

This will unlock the potential benefits of this generalized type of quantum circuit by exploiting more compact representations for quantum states and operations on mixed-dimensional systems. In the following sections, introduce decision diagrams and their benefits, and describe in detail our proposed representations and required manipulation algorithms.

4.2 Mixed-Dimensional Decision Diagrams

A key aspect of developing an efficient simulator is to conquer the correspondingly resulting exponential complexity for as many cases as possible. In the past, decision diagrams have been proven to enable efficient representation of exponentially-sized data in many cases [98]–[102], [105], [130], [145], [146]. In essence, they achieve their compactness by exploiting redundancy in data they represent. This section introduces the decision diagrams for representing quantum states and operations of mixed-dimensional systems.

A *Decision Diagram* (DD) is a *Directed Acyclic Graph* (DAG), composed of nodes and directed edges. The nodes are organized in levels, with each level representing one qudit. The edges represent the connection between the qudits and have additional information annotated to them. In this section we are building upon the intuition given by Section 2.3.1.

4.2.1 Quantum States

The general idea of using decision diagrams to represent quantum states is based on repeated decompositions of the corresponding vectors. More in particular in the qudit case, consider an n-qudit quantum state $q_{n-1}, q_{n-2}, \ldots, q_0$, where q_{n-1} is arbitrarily designated the "most sigificant qudit". This state is split

into equally-sized parts based on the dimensionality of qudit q_{n-1}. Each part is represented by a successor node which also encodes the decision for the value of q_{n-1}. The splitting procedure is then repeated for each part until the individual complex entries in the original vector are reached. During this process, the complex amplitudes of each basis state are stored in the edge weights and equal sub-vectors (up to a complex factor) are represented by the same nodes. To guarantee canonicity, the nodes are normalized such that the sum of squared magnitudes of out-edges of a node add up to one. The resulting decision diagram has a *root node* q_{n-1}, at least one node for each further q_i, and finally a single *terminal node*, which does not have any successor. The decisions on the value of the qudits is recorded so it can be reconstructed. Reconstructing the amplitude for individual basis states is achieved by accordingly traversing the decision diagram and multiplying all edge weights along the path. An example illustrates the idea.

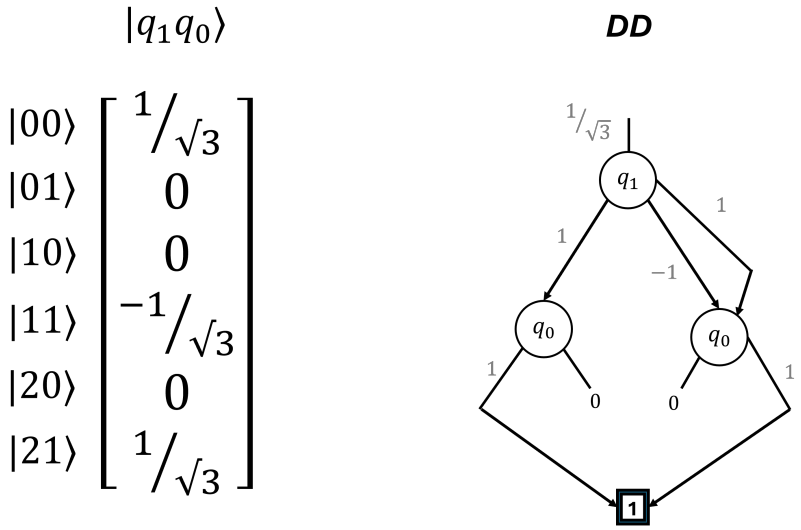


Figure 4.2 A state vector of a qutrit-qubit entangled state and the corresponding DD representation.

Example 13. *Consider the quantum state* $\frac{1}{\sqrt{3}}(|00\rangle - |11\rangle + |21\rangle)$ *in a qutrit-qubit system. Figure 4.2 gives both, the representation as vector and as decision diagram. The vector has 6 entries due to the product of the local dimensionalities:* 3 *for the qutrit and* 2 *for the qubit. The root node* q_1 *has 3 outgoing edges, one for each possible level in qutrit. The nodes in the second level represent the qubit and have 2 edges each. The second and the third edge of the root node point to the same qubit node, due to the exploitation of redundancy. The* q_0 *nodes point to the* terminal node*.*

Finally, in order to retrieve an amplitude of a basis state, we start by multiplying the weights along the path composed of the basis state we want to retrieve. In the of case $|00\rangle$*, we compose* $1/\sqrt{3} \cdot 1 \cdot 1$*, corresponding to the weights of the root node, the edge* 0 *of* q_1*, and the edge* 0 *of* q_0*. In case of the bitstring* $|11\rangle$*, we multiply* $1/\sqrt{3} \cdot -1 \cdot 1$*, corresponding to the weights of the root node, the edge* 1 *of* q_1 *and the edge* 1 *of* q_0*.*

4.2.2 Quantum Operations

The representation of quantum operations as decision diagram follows a similar scheme to that of quantum states. Instead of splitting the vector, for quantum operations the corresponding matrix is split into equal parts depending on the dimenionality of the "most significant qudit", e.g., $3 \times 3 = 9$ parts in case of a qutrit. This scheme is again applied until the individual complex numbers in the matrix are reached. Equal sub-parts up to a complex constant are recognized and stored by the same node in the decision diagram to achieve compactness and canonicity. Again, an example illustrates the idea.

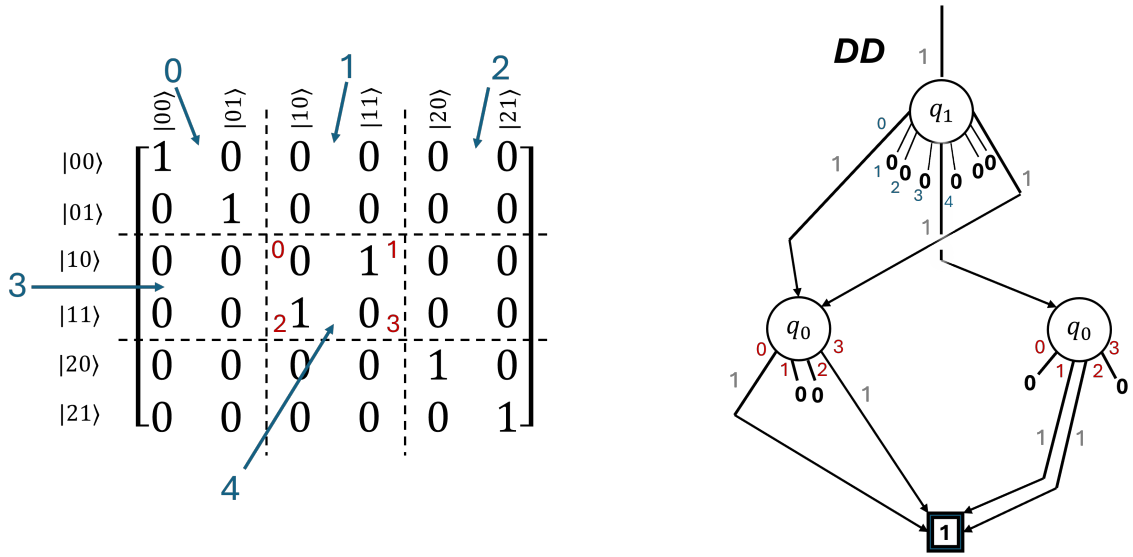


Figure 4.3 A unitary matrix applying the a CEX, controlled on the level $|1\rangle$ of a qutrit applied to $|0\rangle$ and $|1\rangle$ of a qubit.

Example 14. *Figure 4.3 depicts the matrix U for a controlled-on-1-NOT (CEX) between a qutrit and a qubit, consequently the matrix has dimension 6×6. This operation, like most controlled single-qudit operations, is sparse and has high redundancy leading to a very compact decision diagram. Further, the all-zero sub-matrices are immediately represented by* zero stubs *from the root node. Therefore, only three out of the nine edges of the root node representing the qutrit point to a non-zero successor. The first and last edge of the root node, point to the same node that represents the pattern for the identity matrix, while the other non-zero edge in between point to a node that represents the qubit Pauli-X.*

Similarly to the decision diagrams for quantum states, the individual elements of the matrix are retrieved by following a path and multiplying the complex edge weights on this path. Considering the element for mapping $|00\rangle$ to itself (i.e., the element in the upper left corner), we multiply $1 \cdot 1 \cdot 1$, corresponding to the weights of the edge to the root node and following the left-most edges from the following nodes.

4.2.3 The Benefits of DDs

Decision diagrams have proven to be an efficient and compact data structure for exponentially-sized information in many cases in classical design automation and they have shown promising result for design automation in the quantum domain thus far. They exhibit multiple advantages, especially for quantum circuit simulation, such as the following:

- Sub-matrices that contain all-zero elements are represented with a zero-stub, i.e., a single edge of weight zero.
- Decision diagrams are canonical and, therefore, guarantee exploiting redundancies in the data structure.
- During operations like multiplication and addition shared nodes might be encountered more than once. This enables caching of results and, therefore, an improved run time.
- The Kronecker product between two decision diagrams requires only the reassignment of pointers from the terminal node of the first DD to the root node of the second (possibly followed by renormalization).
- Decision diagrams nicely model the locality of a single qudit. The number of edges at each levels matches the dimensionality of the qudit.

These advantages of decision diagrams make them promising candidates for quantum circuit simulation.

4.3 Implementation of Mixed-Dimensional Decision Diagrams

This section illustrates important aspects of the implementation of the previously presented decision diagrams. More precisely, we cover the concepts and methods concerning the construction of decision diagrams and illustrate how the data structure is exploited to perform the basic operations for simulating quantum operations, i.e., multiplication and state measurements. The details on the open-source implementation and how to use it will be covered in *Part* IV.

4.3.1 Representation of States and Operations

As discussed previously, decision diagrams promise to compactly represent quantum states and quantum operations. This, so far, rather abstract discussion is accompanied by an efficient implementation that can handle the representations without intermediate steps that involve the full vectors or matrices. There are two elements that guarantee the efficiency of the implementation, and the first element is the direct construction of quantum states limited to basis states [1], instead of constructing decision diagrams representing quantum states from the corresponding vector, which require exponential memory. This kind of quantum state can be described linearly in the number of qudits. For the construction of the decision diagram, one node for each qudit is required. Building the state from a bottom-up approach will encode the state of each qudit in a single node, connected via the edges (whose edge weights encode the concrete state). By convention, the initial quantum state in simulation is the all-zero state, which has the nodes in the decision diagram only connected by the left-most edges.

The second key element in the implementation is the efficient construction of decision diagrams representing quantum operations: the construction procedure is shown in Algorithm 1. For the proposed simulator, we considered local operations with arbitrary controls where the matrix for the local operation is the only one that is ever actually kept in memory. In the construction, each qudit is again considered in a bottom-up fashion, i.e., from q_0 to q_{n-1}. Each qudit is either (i) the target qudit, (ii) a control qudit, or (iii) not part of the considered operation. For the target qudit, a node with edges holding the values of the local operation is created. For control qudits, if the edges that represent a mapping where the operation is applied are pointing towards the operations, the remaining edges will point to the identity operation. This is known in the process of the construction. However, unlike qubit systems where controls are limited to $|0\rangle$ and $|1\rangle$, the controls can be on arbitrary levels, therefore constructing controlled operations becomes increasingly complex. For qudits that are not part of the operation, the corresponding edges represent the identity operation on these qudits.

To further optimize the construction process, special patterns in the sub-matrices are identified. These patterns include the identity matrix, symmetric, and transposed matrices. Especially for the identity matrix, sub-graphs are stored in a lookup table and referenced within the decision diagram when needed. This approach avoids the repeated construction of identity operations and speeds up normalization routines. Additionally, caching identity patterns helps to efficiently construct decision diagrams for rotations of subspaces within the Hilbert space, where matrices mostly consist of ones with only a few entries representing the actual operation.

The key difference to previous implementations of decision diagrams is the added capability of dynamically creating decision diagrams with different dimensions for each qudit. More precisely from the implementation perspective, the number of out-edges of a node can be changed to accommodate the given dimensionality. This marks a significant departure from the fixed and static structures of previous types of decision diagrams [98]–[101], [105], [130], making it adaptable for future use cases where temporary expansion

Algorithm 1 Make Gate DD

Require: Matrix Operation U, lines L, target line t, controls $c \subset C$
Ensure: root edge e of Gate DD

```
e_U = U entries as edges
for l in L, l < t do
    Allocate e_L = l.size() zero edges
    for e_i in e_U do
        if l is ctrl, w/ ctrl c and e_i is diag then
            e_{L_c} ← e_i
            for e_{L_j} in e_L, j ≠ c do
                if e_{L_j} is diag then e_{L_j} ← I
                else e_{L_j} ← zero
        else if l is ctrl, w/ ctrl c and e_i not diag then
            e_{L_c} ← e_i
        else                                          ▷ Line is not a control
            for e_{L_j} in e_L do
                e_{L_j} ← e_i
    e_i ← e to norm node w/ successors e_L
e_U ← e to norm node w/ successors e_i                ▷ Target line
for l in L, l > t do                                  ▷ Variables above the target
    Allocate e_L = l.size() zero edges
    for e_j in e_L do
        if l is ctrl, w/ ctrl c and e_j is diag then
            if j = c then e_j ← e_U
            else e_j ← I
        else                                          ▷ Line is not a control
            if e_j is diag then e_j ← e_U
    e_U ← e to norm node w/ successors e_L
```

of the Hilbert space of a qudit could be a crucial component for shorter circuits. The implemented data structure is closer to the behavior of nature than ever before.

4.3.2 Applying Quantum Operations

Given the construction of quantum states and operations detailed in the previous section, this section covers the part of the implementation that applies the operations to the states. To this end, we consider the Kronecker product, multiplication of decision diagrams, and measurement as the essential parts of the implementation.

Kronecker Product

The first quantum operation considered is the Kronecker product for decision diagrams. For matrices, it is defined as

$$A \otimes B = \begin{bmatrix} a_{0,0} \cdot B & \cdots & a_{0,D-1} \cdot B \\ \vdots & \ddots & \vdots \\ a_{D-1,0} \cdot B & \cdots & a_{D-1,D-1} \cdot B \end{bmatrix}.$$

This means that each element of A is replaced by the element $a_{i,j} \cdot B$. Moreover, on matrices the Kronecker product is computationally expensive, since each of the elements of the exponentially-sized matrices has to be accessed at least once.

However, in decision diagrams, the Kronecker product is efficient. The terminal node of a decision diagram representing A is replaced by the root node of B by means of changing the corresponding pointers. Afterwards, for normalization, the weight of the previous edge pointing to the root node of B is multiplied to the root edge to A. This process is detailed in Algorithm 2 and illustrated in Figure 4.4.

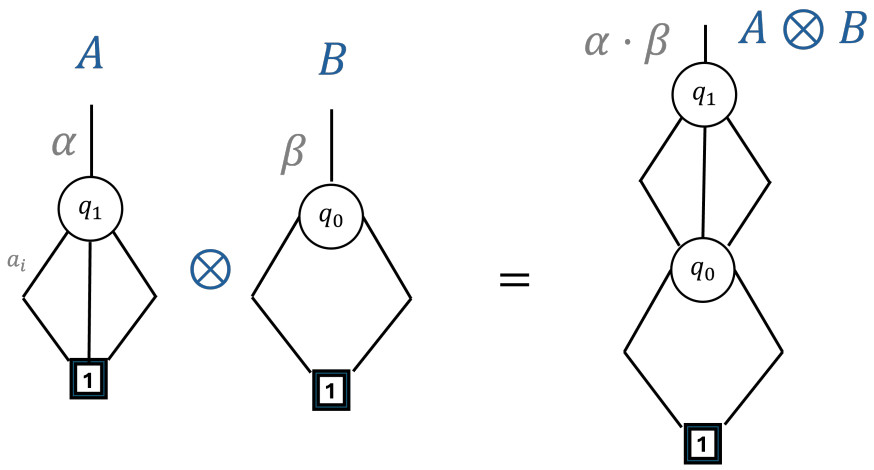


Figure 4.4 Kronecker product for decision diagrams

Algorithm 2 Kronecker Product

Require: root edge x, root edge y
Ensure: root edge e of $x \otimes y$
  if x or y is zero **then return** zero
  if x is scalar **then return** $y.weight \times x.weight$
  if $LookUp(x, y)$ **then return** $cached(x, y)$
  Allocate e_N = $x.node.size()$ zero edges
  for e_i in e_N **do**
    $e_i \leftarrow$ *Kron*$(x.node.e_i, y)$
  $e_K \leftarrow e$ to norm node w/ successors e_N
  $e_K.weight \leftarrow e_K.weight \times x.weight \times y.weight$
  return e_K

The implementation of the Kronecker product uses a recursive approach that traverses the decision diagram structure; hence, the computational complexity of the operation is linear in the nodes of the left-hand side operand.

Multiplication and Addition

As discussed in Section 4.1.2, multiplication and addition are the quintessential operations necessary to conduct quantum circuit simulation. Given a quantum state $|\psi\rangle$ (with vector representation ψ) and a unitary operation U, the operation is applied by:

$$\psi_i' = \sum_{k=0}^{D-1} u_{i,k} \cdot \psi_k,$$

where D is the dimension of the vector and $u_{i,k}$ (ψ_k) represents an element in the matrix (vector). We can use the recursive conceptual decomposition of the matrix (vector) represented by the respective DD, but this time to perform the multiplication. We use

$$\begin{aligned}\psi' = U \cdot \psi &= \begin{bmatrix} U_{0,0} & \cdots & U_{0,D-1} \\ \vdots & \ddots & \vdots \\ U_{D-1,0} & \cdots & U_{D-1,D-1} \end{bmatrix} \cdot \begin{bmatrix} \psi_0 \\ \vdots \\ \psi_{D-1} \end{bmatrix} \\ &= \begin{bmatrix} \sigma_0 \\ \vdots \\ \lambda_0 \end{bmatrix} + \cdots + \begin{bmatrix} \sigma_{D-1} \\ \vdots \\ \lambda_{D-1} \end{bmatrix}\end{aligned}$$

The algorithm will recursively follow corresponding paths in the DDs for the matrix and the vector, and apply the operation of multiplication between the sub-matrices and sub-vectors, as detailed in Algorithm 3 and Algorithm 4. All the intermediate state vectors can be added recursively, i.e.,

$$\psi' = \begin{bmatrix} \sigma_0 + \cdots + \sigma_{D-1} \\ \vdots \\ \lambda_0 + \cdots + \lambda_{D-1} \end{bmatrix}$$

The order and structure of the sub-multiplications and sub-additions follow the structure of the decision diagram. Consequently, the complexity is bound to the decision diagram in terms of number of nodes. However, the reduction of the nodes allows to cache intermediate results. Finally, the reduction of DDs potentially delivers an exponential improvement in terms of memory complexity, as well as in terms of time complexity by terminating early sub-computations already performed. The final step of multiplication is, naturally, the normalization of the decision diagram. The procedure is recursive and included in the steps described in Algorithm 4.

The multiplication and addition operations have been optimized to account for the dynamic structure of the decision diagrams. Unlike previous versions of the data structure where the number of edges was fixed and known beforehand (i.e., pointers in an array), the current version uses a vector with variable-sized edges (pointers), which allows for greater flexibility. Despite the added overhead of checking the size and iterating over each edge, the implementation remains relatively efficient compared to previous fixed versions of the data structure.

DD Optimizations in Caching Mechanisms

In Figure 4.5, we see the multiplication between a matrix and vector DD, where the problem is divided in two, and the multiplication is solved for the right branch and left branch separately. It is interesting to note that when we multiply two branches that point to two nodes that have been multiplied before the result is retrieved by a cache that kept the result of previous computations, while when two branches are multiplied and one points to the zero terminal node, the computation is directly returned to be zero.

Algorithm 3 Multiplication

Require: root edge x, root edge y
Ensure: root edge e of $x \times y$
  if x or y is NULL or $x.weight$ or $y.weight = 0$ **then return** zero
  if $Terminal(x)$ and $Terminal(y)$ **then return** Terminal e, w/ $e.weight = x.weight \times y.weight$
  if $LookUp(x, y)$ **then return** $cached(x, y)$
  if x is I **then return** y
  if y is I **then return** x
  Allocate $e_N[R]$ zero edges ▷ Edges for Results
  for i in R **do** ▷ Rows
    for j in C **do** ▷ Columns
      for e_k in e_N, $k = R * i + j$; **do**
        $e_k \leftarrow e_k + Mult(e_x[R \cdot i + k], e_y[j + C \cdot k])$
  $Cache\ e_K \leftarrow e$ to norm node w/ successors e_N
  $e_K.weight \leftarrow e_K.weight \times x.weight \times y.weight$
  return e_K

Algorithm 4 Addition

Require: root edge x, root edge y
Ensure: root edge e of $x + y$
  if $Terminal(x)$ **then return** y
  if $Terminal(y)$ **then return** x
  if $x.node = y.node$ **then return** e, w/ $e.weight = x.weight + y.weight$ and $e.node = x.node$
  if $LookUp(x, y)$ **then return** $cached(x, y)$
  Create e_N edges with $e_N.size() = x.nodes.edges.size$
  for $i = 0 \dots e_N.size() - 1$ **do**
    $e_{y,next} \leftarrow e_{yi.node}$; $e_{y,next.weight} \leftarrow y_{weight} \times e_{y_i.weight}$
    $e_{x,next} \leftarrow e_{xi.node}$; $e_{x,next.weight} \leftarrow x_{weight} \times e_{x_i.weight}$
    $e_j = add(e_{x,next}, e_{y,next}), i = j$
  return e to norm node w/ successors e_N

Measurement

The measurement can be interpreted as a global measurement since the outcome of the simulation is the state vector of the system. Decision diagrams enable a measurement procedure that is linear in the number of qudits.

The process of measuring from the decision diagram requires one traversal of the decision diagram in a top-to-bottom manner, starting from the root node. At each node, the squared magnitude of each edge weight are the probability that this edge should be followed. After following the edge to the next node, this procedure is repeated until the target node is reached. The corresponding basis state is then given by the path through the decision diagram (the index of each edge gives the corresponding level, starting with zero) and the product of the edge weights on the path gives the corresponding amplitude.

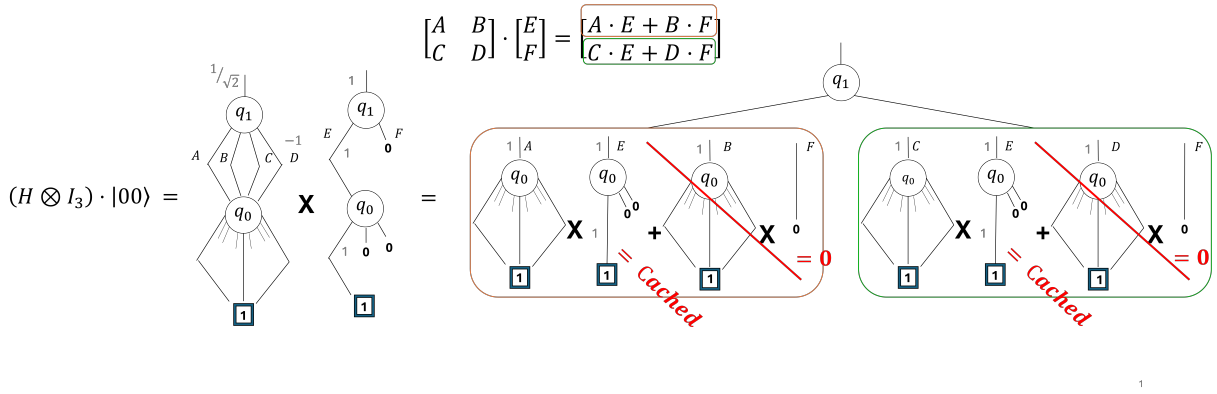


Figure 4.5 Exploitation of the structure of the DDs for optimizing the operations of multiplication by using caching mechanisms.

4.4 Experimental Evaluation

We implemented the proposed approach and evaluated its applicability. The details on how to use the implementation and the the open source code will be covered in *Part* IV.

The evaluation was performed on a server running GNU/Linux with an Intel Xeon W-1370P (running at 3.6 GHz) and 128 GiB main memory. The implementation is written in C++ and was compiled with GCC 9.3.0. To demonstrate the capabilities, a set of three different algorithms with multiple instances was considered more precisely:

- Mixed W-States [147] given as states $\frac{1}{\sqrt{n}}(|0\ldots01\rangle+|0\ldots010\rangle+|10\ldots0\rangle)$. Qu*b*it W-states embedded in prime-dimensional qu*d*its.
- *GHZ States* [148] were considered for n qutrits as $\frac{1}{\sqrt{3}}(|0\rangle^{\otimes n}+|1\rangle^{\otimes n}+|2\rangle^{\otimes n})$.
- Random Circuits built from randomly selected local operations (Hadamard and Givens rotations) and entangling operations (CEX and controlled Clifford operations).

The results are summarized in Table 4.1. The first column lists the name of the benchmark. The second column gives the total number of qudits where as the following four columns give the number of qudits for each level, i.e., the number of qubits (2), qutrits (3), ququarts (4), and ququints (5). In the following three columns, the number of operations, number of nodes, and number of distinct complex numbers in the decision diagram of the final state after the simulation are listed. Finally, the last column lists the run time of each instance of the benchmark.

For the algorithms Mixed W-State and GHZ state, Table 4.1 nicely confirms the efficiency of the proposed approach and the corresponding implementation. The simulation for these algorithms completes in less than a second with a low number of nodes and distinct complex numbers, as one might expect. In contrast, the random circuits contain little to no redundancy to be exploited by the decision diagrams and, therefore, can be considered to show the worst-case behavior. Still, the implementation can simulate these circuits with, for example, 8000 operations on 8 qudits within around 4 h, which, given the individual number of levels, are comparable to 14 qubits. The simulator is remarkable in its ability to deliver reliable results for randomized circuits, also when those are comparable to a larger 15-qubit system. Although the simulation process largely terminates within two days, this time frame is a testament to the efficiency and accuracy of the simulator's algorithms. These results confirm the efficacy of the simulator as a design tool for mixed-dimensional quantum circuits, for a large set of mixed-dimensional architectures and benchmarks.

This new mixed-DD-based simulator is proven to be a food ad-hoc solution for verifying and validating mixed-dimensional quantum circuits.

4.5 Conclusions

In this chapter, we introduced a type of decision diagram capable of handling the simulation of mixed-dimensional quantum circuits. More precisely, the proposed decision diagrams encode the dimensionality of a qudit in the number of out-going edges of a node, enabling a dynamic handling of the dimension and fast adaption to the restrictions imposed, e.g., after calibration of a qudit system. This compact representation is the basis for the corresponding implementation, which enables utilization of the method in real world context. The experimental evaluation confirms the efficacy for the selected set of benchmarks, ranging from easy circuits such as the GHZ state to random circuits, which show the worst-case behavior. The details on how to use the implementation and the the open source code will be covered in *Part* IV.

Table 4.1 Evaluations of MiSiM on Mixed-Dimensional Benchmarks

Benchmark	#Qudits	#Qudits with Dimension				#Operations	#Nodes	#Distinct $\mathbb{C}$	Run time
		2	3	4	5				
Mixed W-State	4	2	2	0	0	15	12	9	0.000
	30	15	3	0	12	96	60	64	0.017
	54	2	52	0	0	159	108	112	0.027
	60	8	4	0	48	213	120	140	0.169
	90	15	63	0	12	276	180	198	0.103
	90	2	80	0	8	273	180	195	0.101
	102	75	2	0	25	315	200	207	0.111
	108	8	100	0	0	321	216	229	0.102
GHZ State	5	0	5	0	0	8	14	5	0.000
	10	0	10	0	0	18	29	5	0.000
	30	0	30	0	0	58	89	5	0.003
	60	0	60	0	0	118	179	5	0.011
	120	0	120	0	0	238	359	5	0.043
	128	0	128	0	0	254	383	5	0.049
Random	2	0	0	2	0	2000	6	68 903	0.021
	2	0	2	0	0	2000	5	40 835	0.015
	2	0	0	0	2	2000	7	112 918	0.068
	2	0	1	1	0	2000	5	52 451	0.016
	2	0	1	1	0	2000	5	52 451	0.016
	3	2	1	0	0	3000	10	66 575	0.021
	3	0	0	2	1	3000	27	531 995	0.172
	3	0	0	2	1	3000	27	545 173	0.184
	3	1	0	2	0	3000	12	168 915	0.048
	3	1	0	2	0	3000	14	227 513	0.069
	4	1	2	0	1	4000	67	774 865	0.339
	4	1	1	1	1	4000	44	870 383	0.380
	4	2	2	0	0	4000	22	239 247	0.069
	4	0	1	1	2	4000	107	2 651 118	4.172
	4	2	1	0	1	4000	41	429 614	0.153
	4	0	0	2	2	4000	106	3 141 654	6.564
	4	0	2	1	1	4000	53	1 299 120	0.828
	5	0	0	3	2	5000	507	16 771 562	231.361
	5	1	1	2	1	5000	214	4 578 367	13.622
	5	1	1	2	1	5000	161	3 287 282	6.592
	5	2	0	1	2	5000	137	3 181 180	6.258
	5	2	1	0	2	5000	101	2 319 028	3.094
	5	2	0	1	2	5000	164	3 435 001	7.242
	5	1	1	1	2	5000	217	5 732 896	22.778
	6	2	1	1	2	6000	434	11 781 434	105.238
	6	2	2	1	1	6000	486	7 518 639	38.225
	6	2	1	0	3	6000	437	13 889 217	159.516
	6	4	2	0	0	6000	137	1 324 892	0.959
	7	3	1	2	1	7000	1308	23 540 630	390.539
	7	3	1	0	3	7000	2447	42 457 590	1275.100
	7	3	2	0	2	7000	577	16 793 344	227.600
	7	2	2	1	2	7000	1301	41 679 061	1407.100
	8	5	2	1	0	8000	572	11 114 401	85.309
	8	2	3	1	2	8000	3902	135 137 166	14 972.200
	8	2	1	2	3	8000	19886	432 117 869	170 405.000

The column "#Distinct $\mathbb{C}$" shows the number of different complex numbers in the decision diagram.

5 Refining and Simulating Mixed-Dimensional Quantum Noise

The quantum computing platforms that we reviewed in Section 2.1 are affected by frequent errors due to the inherently delicate nature of quantum systems [18]. Despite advances in error mitigation efforts, these errors continue to be the major challenge in quantum computing. Given the benefits of classical simulation studied in *Chapter* 4, it is essential to incorporate these errors during quantum circuit simulations to accurately predict an algorithm's performance on real quantum hardware and validate the error modelling of a mixed-dimensional quantum device.

Noise models for mixed-dimensional systems can be described through quantum mechanical formalisms, such as quantum channels and mixed states [1], although they are too general to be used for validating experiments, as shown in [74]. Advanced data structures such as tensor networks reviewed in Section 2.3.2 and decision diagrams reviewed in Section 2.3.1 have been shown to reduce the enormous complexity of simulating classical quantum circuits in numerous practically significant cases, forming a fundamental basis for efficient quantum circuit simulation. However, the extensive framework of error modeling, that requires computationally intensive formalisms like density matrices, further complicates the already exponentially difficult task of simulating quantum circuits.

This chapter explores the fundamentals and methods for simulating noisy quantum circuits, particularly emphasizing noise modeling in systems with mixed dimensions, formalizing for the first time a noise model for mixed-dimensional systems in line with the state of art machines [9]. A new specification of the depolarizing noise channel is presented, which aims to approximate the noise in mixed-dimensional systems with good precision, even in shallow-depth regimes. Subsequently, we provide an implementation that could allow us to estimate the actual impact of errors on quantum computations by means of Monte-Carlo simulations, allowing us to calculate empirical average effects across multiple runs. We employ a stochastic error model, assuming errors occur randomly in each simulation run. Although the numerical results are beyond the scope of the chapter, it offers a conceptually sound and mathematically robust method for classically simulating noisy quantum computations, which can be easily integrated into existing tensor network and decision diagram simulators, as detailed in *Part* IV.

The chapter is structured as follows. Section 5.1 reviews the possible quantum computing errors. In Section 5.2.3, we outline the concept of the proposed solution, and in 5.3.1, we will illustrate the mathematical modeling used for describing the stochastic simulation. The full implementation and the the open-source code will be covered in *Part* IV.

5.1 Sources of Noise in Quantum Computations

This section briefly overviews the error types and noise prevalent in quantum computing systems, focusing on their impact on single-qubit states, which have been covered in Section 2.2.1. In quantum computing, errors can be broadly divided into coherent and incoherent types. Coherent errors occur when faulty or unintended unitary operations cause changes in quantum states, often due to gate calibration inaccuracy or unwanted qubit interactions. In contrast, incoherent noise, or decoherence, involves irreversible state alterations. Recognizing these noise sources is important for the design of mixed-dimensional quantum

devices, mitigating errors, improving circuit designs, and establishing quality metrics. Additionally, we have effects of decoherence that arise from unavoidable interactions between qudits and the environment, affecting coherence time and gate precision and thus limiting operations. In multi-qubit systems, noise impacts entangling fidelity and necessitates error correction due to correlated errors. In Figure 5.1, quantum states are depicted as vectors that reside on the surface of a unit sphere called *Bloch sphere*. For qubits, states $|0\rangle$ and $|1\rangle$ are located at the poles. This practical representation helps in understanding the impacts of quantum noise. We can examine in order some of the significant types of noise and their effects on quantum states.

Dephasing When a qudit interacts with the environment, a phase flip effect might occur. This leads to an error called *phase flip error* or *T2 error*. On a qubit this happens to be a phase-flip. On qudits the phase-flip happens for each dimension.

Amplitude Damping A qudit in a high-energy state ($|2\rangle$ or $|1\rangle$) tends to relax into a low energy state ($|1\rangle$ or $|0\rangle$). After a certain amount of time, qudits in a quantum system eventually decay completely to $|0\rangle$, the minimal energy state. This error is called *amplitude damping error* or *T1 error*.

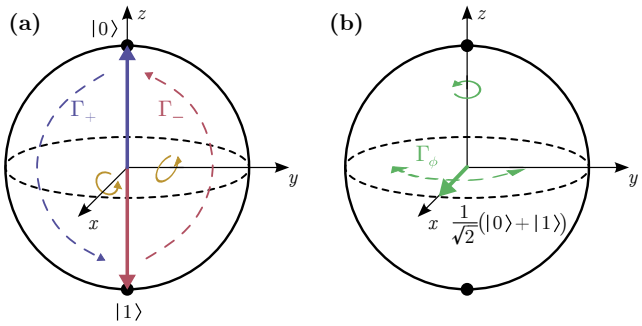


Figure 5.1 Bloch-sphere representation of amplitude dumping on the left, and pure dephasing on the right [149].

Depolarizing Noise This type of incoherent noise is distributed uniformly around the Bloch sphere (meaning each state has an identical likelihood of encountering bit-flip and phase-flip errors), and a qubit will eventually experience decoherence, leading to a total loss of quantum information. This phenomenon is referred to as depolarizing noise, as it causes the Bloch vector to depolarize and move towards the center of the Bloch sphere.

Stochastic Pauli Noise In various systems, noise exhibits a bias, resulting in the potential for random bit or phase flips around different axes to occur at varying frequencies. This can be represented through random Pauli errors, where each specific Pauli error type, represented by a Pauli operation, has its own probability of happening.

Special Cases Several types of noise are difficult to categorize and have particular effects, like leakage, which refers to the phenomenon in which a qubit becomes excited beyond the computational basis, a process that can occur due to stochastic thermal fluctuations or coherent driving. Within the framework of qubit computations, leakage is frequently characterized as a non-Markovian event, given that it can show temporal correlations spanning several gates or cycles.

5.2 Motivation

We had so far the chance to elaborate on the basics of quantum computing (*Chapter* 2) and on simulation *Chapter* 4 with elaborate techniques and data structures for mixed-dimensional systems. The previous section (Section 5.1) introduced the notion of errors in quantum computations with some intuition of the effects on the quantum state. Here, we describe some examples of efforts made to predict the effect of errors on quantum computations through simulation.

5.2.1 Problem

In the previous chapters, we described the challenges of simulating quantum circuits, where the objective is to acquire a final quantum state, denoted as $|\psi\rangle$, corresponding to the output of the quantum algorithm run on the real device. Although the issue appears straightforward, it is significantly complicated by the curse of dimensionality (Section 4.1.2). On top of that, reaching the final state alone is insufficient for determining the error characteristics of the quantum systems used; in fact, A single quantum state cannot describe complex phenomena of superposition such as *mixed-states* [1]. More elaborate mathematical tools such as *density matrices* and *quantum channels* allow a precise and complete representation of the information.

For example, the density matrix offers a representation of the statistical state of the quantum system expressed by the formula

$$\rho = \sum_i p_i |\psi_i\rangle\langle\psi_i|$$

where p_i indicates the probability that the quantum system is in state $|\psi_i\rangle$. The pure quantum states $|\psi_i\rangle$ all together then define the mixed state represented by the density matrix ρ. Note that due to the exponential growth of the state, with size D, the dimensions of the matrix will also exponentially increase with dimensions $D \times D$. This method for tracking the effects of noise transforms the already complex problem of classical simulation into one of even greater complexity.

Furthermore, studying the influence of errors within simulations is a remarkable challenge, as these effects manifest randomly, with probability depending on the quantum hardware to be simulated. Consequently, rather than obtaining a single potential final state, simulations incorporating errors generate a range of possible output states, a product of the specific error effects introduced. For this reason, there are continuous efforts in modelling the noise source and its effects, that will be used to drive this type of simulation.

The challenge of simulating noisy mixed-dimensional systems consists of the same set of computational and modeling problems as other quantum computers, but the modeling of quantum noise in classical quantum circuit simulations falls behind. Formalisms like the description of the depolarizing noise channel, are theoretically correct but struggle to predict the effects of errors in quantum circuits with different depth regimes for the mixed-dimensional case. In fact, while this approximation works well for both deep and shallow qubit circuits, this description is quite inaccurate for mixed-dimensional architectures where the gate sets can often operate on subspaces with both entangling and local operations and which have different error rates, as shown in [74]. The empirical results illustrate different behaviors from what the theory expects. Approximating the depolarizing noise channel with the application of qudit Pauli noise at each operation on the whole qudit space appears to be a too generic approach since the operations would manipulate subspaces that should be idle.

The challenges associated with simulating noisy mixed-dimensional quantum circuits are related to the correctness and to complexity of the simulation, hence in this chapter, we focus on two primary issues:

- The initial concern is the necessity for more precise approximations of the mixed-dimensional depolarizing noise channel, capable of operating across various regimes, in alignment with experimental results.

- The secondary aim is to develop a computationally efficient technique to apply the newly proposed model and to execute simulations of noisy quantum circuits involving mixed-dimensional systems.

We need to solve these problems to create reliable simulations that will allow to analyze in the long term the execution of quantum circuits of different sizes under realistic assumptions of noise. The new approach will leverage the performance of current data-structures and methods with minimal implementation cost, while exploiting the progress made in modeling noisy qubit circuits.

5.2.2 Present Results

Here, we review existing methods for simulating noisy quantum circuits. In the study [150], researchers utilize stochastic ensembles of quantum circuits, incorporating random Kraus operators into the original quantum gates to model random errors within quantum channels. They demonstrate that the stochastic simulation error remains relatively low, even when simulating many qubits in comparison to the density matrix method. The simulations were performed through the use of tensor networks, previously discussed in Section 2.3.2. The results showcased simulations with up to 30 qubits, achieving less than 1% error compared to density matrix simulations, allowing for embarrassingly parallel computations.

Other works [151]–[153] used decision diagrams to simulate noisy quantum circuits. In [151], decision diagrams are used similarly to [150], where random Kraus operators are applied stochastically to quantum gates to mimic random errors and model quantum channels. Decision diagrams, along with parallel executions, significantly reduce resource requirements. In [152], [153], this approach is adapted in order to perform the deterministic simulations of noisy quantum circuits based on the density matrix formalism, where the matrices are represented through decision diagrams.

These examples illustrate several strategies and data structures used to address the issue of noisy quantum circuit simulation. However, the studies presented thus far do not address efficient simulation of mixed-dimensional quantum systems. They do not focus on modeling noise in such systems but on qubit-based proof-of-concept simulations.

5.2.3 Proposed Solution

In this section, we propose a solution to the noisy quantum circuit simulation of arbitrary mixed-dimensional quantum systems to enable realistic simulations of quantum circuits in shallow and deep regimes.

To this end, the proposal is divided into two parts:

1. the first part is the proposal of a new noise model, a depolarizing noise model adapted for the description of the effect of operations manipulating qudit subspaces,

2. the second part is a proposal of stochastic Monte-Carlo simulations for mixed-dimensional quantum circuits, with an analysis of
    - why the model could be appropriate for our simulation goals,
    - how it could benefit runtime in the case of decision diagrams based simulation.

The noise model introduced in our solution is based on the depolarizing noise model, which is approximated in the qubit case by stochastic Pauli noise. While the depolarizing noise model for qudits is too general and acts on the entire single-qudit space, a more fine-grained approach is needed. To address this, we have modeled the subspace noise through the qubit depolarizing noise model, treating the two-dimensional subspace as an embedded qubit within the larger Hilbert space. The noise of each operation is then approximated by a subspace operation that corresponds to the qubit Pauli operators embedded in

the subspace. The subspace dephasing noise is modeled as a phase flip on each dimension independently. This approach allows us to define noisy behaviors for shallow circuits with numerous idle subspaces while approximating the qudit depolarizing noise in deep circuits. The qudit depolarizing noise channel remains valid for all operations acting on the full space of a single qudit. This model remains valid, produces results in principle consistent with experiments, and preserves formal correctness.

The second step involves implementing a proof-of-concept stochastic and parallel simulation of quantum circuits. The simulation follows the Monte-Carlo paradigm, where individual simulations are processed in an embarrassingly parallel fashion, with noise applied after each gate in the circuit according to specified probabilities. The innovative aspect of this approach lies in its software engineering: the instantiation of a noisy quantum circuit is decoupled from its simulation. For each simulation run, a noisy quantum circuit is generated by adding either embedded Pauli operations or qudit Pauli operations, and a simulation engine then processes this circuit. The Monte-Carlo simulation approach is chosen particularly for its approximating capabilities of the chosen observables. This implementation of the noise model has specific numerical implications and advantages for decision diagrams, which will be examined in a subsequent section.

In the following section, we introduce the modeling of mixed-dimensional noise and the implementation of Monte Carlo simulation. Our proposed solution represents the first implementation of a mixed-dimensional noise simulation approach.

5.3 Mixed-Dimensional Noisy Quantum Circuit Simulation

In this section, we present the proposed mixed-dimensional noise model to improve the simulation of noisy quantum circuits.

5.3.1 The Mixed-Dimensional Noise Model

Developments in the physical realization of quantum computers show significant improvements in the coherence times of qudits [9], improving the “lifetime” of qudits before decaying to lower energy states, and reducing the frequency of phase flip errors. However, more errors have to be studied and identified.

Depolarizing Channel

More specifically, *depolarizing noise* is the process that can transform into a completely mixed state, a quantum state ρ with some probability p. In practice, the depolarizing noise acts isotropically around the Bloch sphere, which means it is like applying Pauli X, Y, and Z randomly and with an equal probability of occurring for all states. Therefore, in a visual interpretation of the qubit case, the depolarizing noise reduces the length of the Bloch vector with a depolarizing probability p.

If we write the depolarizing noise in a more intuitive way, we get the following formula,

$$\mathcal{E}(\rho) = (1 - p')\,\rho + \frac{p'}{3}(X\rho X^\dagger + Y\rho Y^\dagger + Z\rho Z^\dagger)\,, \tag{5.1}$$

where we take $p' = 3p/4$. In this form, we may interpret a depolarizing noise channel as one in which the qubit experiences an error X, Y, and Z, each with probability $p'/3$, but remains unchanged with probability $1 - p'$.

Example 15. *Consider a 2-qubit state $|\psi'\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$. Suppose that the second qubit in order is getting depolarized with a probability p and with probability $1 - p$, nothing happens and the state remains*

unchanged. The effect is similar to applying I, X, Y, or Z, each with probability $\frac{p}{3}$. The final mixed-state will be composed of

$$\{(1-p,(\tfrac{1}{\sqrt{2}})(|00\rangle+|11\rangle),(\tfrac{p}{3},(\tfrac{1}{\sqrt{2}}(|01\rangle+|10\rangle),(\tfrac{p}{3},(\tfrac{i}{\sqrt{2}}(-|01\rangle+|10\rangle),(\tfrac{p}{3},\tfrac{1}{\sqrt{2}}(|00\rangle-|11\rangle)\}$$

which cannot be represented by a single 2-qubit state.

Dephasing Channel

Dephasing is the loss of phase coherence in a quantum state. It can be observed in the reduction of the off-diagonal components of the density matrix, and in the experiment, dephasing happens when there is a desynchronization between the qubit oscillations and the control field, usually originating from frequency variations. Additionally, in the subspace, the relative phase changes at a rate correlated to the energy difference of the states.

In the Kraus representation, dephasing noise is given by

$$\mathcal{E}(\rho)=\left(1-\frac{p}{2}\right)\rho+\frac{p}{2}Z\rho Z^{\dagger}\,, \tag{5.2}$$

Here, a quantum state ρ undergoes a phase-flip with probability $p/2$ and is unchanged with probability $1-p/2$.

Mixed-Dimensional Noise

After reviewing the two main types of noise that affect quantum computer circuits' execution, we reveal the proposal for modeling noise in the mixed-dimensional case. The operations performed on the whole qudit space are still affected by the depolarizing noise, with the following mathematical description:

$$\mathcal{E}(\rho)=(1-p)\rho+\frac{p}{d^2-1}\sum_{\substack{j,k=0\\(j,k)\neq(0,0)}}^{d-1}X^jZ^k\rho(X^jZ^k)^{\dagger} \tag{5.3}$$

Eq. (5.3) represents the generalization of Eq. (5.1) to higher dimensions. In fact, the explanation and formulas of this section have so far been focused on the qubit case. The equation here describes how the state is left unaltered by the noise with probability $1-p$, and is altered according to the probability $\frac{p}{d^2-1}$, forming a possible ensemble of outcomes. This probability is derived similarly to before, however in this case the probability of an error occurring is divided uniformly over the d^2 possible choice of qudit Pauli operators. The generalized operators are constructed by composing the clock and shift operators described in Section 2.2.2.

Inspired by empirical results [74], we introduce two modifications to the treatment of subspace operations that will make the depolarizing noise model closer to the physics of the devices. The first novelty is the subspace depolarizing noise, which is modeled like in the qubit case but adapted to act only on a subspace spanned by two dimensions at a time; in fact, the model is applied to operations that act in a specific subspace. The physical explanation is that an operation manipulates a specific transition of the qudit that can be naively seen as a two-level physical entity, therefore the depolarizing noise model for the subspace of $|i\rangle$ and $|j\rangle$, while using the practical notation with the p' probability is as follows:

$$\mathcal{E}(\rho)=(1-p')\,\rho+\frac{p'}{3}(X_{i,j}\rho X_{i,j}^{\dagger}+Y_{i,j}\rho Y_{i,j}^{\dagger}+Z_{i,j}\rho Z_{i,j}^{\dagger})\,, \tag{5.4}$$

the Kraus operators are embedded qubit operations in the qudit space, similar to the rotation in Example 6. Numerically, the entries of the operators will belong to the ring of the qubit Pauli group. The dephasing error is qudit systems should be equivalent to the qubit simplified case:

$$\mathcal{E}(\rho) = (1 - p')\,\rho + p' Z \rho Z^{\dagger}\,, \tag{5.5}$$

where the simplified notation explains that with probability p' the state is "shifted" by the qudit Z operation.

The second novelty is the introduction of the subspace dephasing noise that is described as the phase flip on a set of qudit dimensions, according to

$$\mathcal{E}(\rho) = \prod_{k \neq i,j} \left[(1 - p')\,\mathbb{I} + p'\, Z_k(\cdot) Z_k^{\dagger}\right](\rho)\,, \tag{5.6}$$

The equation uses once more a simplified notation where p' is the probability of an error happening. The error has a physical interpretation that is to be conducted to the results in the experiment [74]. Eq. (5.6) shows that each time a subspace operation is applied to a qudit, the remaining $d - 2$ dimensions in the qudit suffer from a slight excitement, which can be conducted to a Stark shift of some sort in the case of ion qudits. Therefore, given an operation on the dimensions $|i\rangle$ and $|j\rangle$, the equation tells us that we can independently apply a phase flip on the remaining dimensions $d - 2$. Although we use Z as notation, the matrix is just the identity with a -1 on the Kth dimension to be flipped; commonly, we refer to these operations as *Virtual Z Rotations* (Vz, or VirtRz) since this type of operations is applied to the frame of the qudit.

The conceptualized noise framework allows for fine-grained control over noise channels, supporting noise on local operations, as well as multi-qudit operations with different effects depending on the gate specification.

Example 16. *Consider a 2-qutrit state* $|\psi'\rangle = \frac{1}{\sqrt{3}}(|00\rangle + |11\rangle + |22\rangle)$*, where a Givens rotation on a subspace is applied to the first qutrit. Suppose that this state might be affected by a local gate error, depolarizing it. With probability* $1 - p$*, the state is left untouched; on the other hand, with probability* p*, the second qutrit in order becomes depolarized. The effect is captured by applying I,* $X_{a,b}$*,* $Y_{a,b}$*, or* $Z_{a,b}$*, each with probability* $\frac{p}{3}$*. This produces an ensemble (or mixture) if we set* a *to 0 and* b *to 2:*

$$\{(1-p, \tfrac{1}{\sqrt{2}}(|00\rangle+|11\rangle+|22\rangle), (\tfrac{p}{3}, (\tfrac{1}{\sqrt{3}}(|20\rangle+|11\rangle+|02\rangle), (\tfrac{p}{3}, (\tfrac{i}{\sqrt{3}}(-|20\rangle+|11\rangle+|02\rangle), (\tfrac{p}{3}, \tfrac{1}{\sqrt{3}}(|00\rangle+|11\rangle-|22\rangle)\}$$

which cannot be represented by a single 2-qutrit state.

5.3.2 Implementing Monte Carlo Simulations

The second part of the proposed solution is the implementation of stochastic simulations based on the model proposed above. When a quantum computer might produce inaccuracies with a particular probability p, we model this error in simulations with the equivalent probability p, and keep the state unchanged with a probability of $1 - p$. Essentially, we decide with a given probability whether to sample specific noise for each operation. The ultimate state is shaped by the decisions made throughout the simulation. This simulation category is termed a *Monte-Carlo* simulation, where the objective is to approximate the evaluation of a function over the actual distribution of states by computing empirical averages from the sampled output states. The method is particularly beneficial because it allows the approximation of some properties of a distribution without reconstructing the entire distribution.

In practice, the output is a collection of states generated, each from a single simulation,

$$|\tilde{\psi_1}\rangle, \ldots, |\tilde{\psi_M}\rangle$$

and from each state, we can estimate many quadratic properties at once. In quantum computing, we can estimate many interesting properties, like the fidelity, via a quadratic function:

$$o_l = |\langle \omega_l | \psi \rangle|^2 \tag{5.7}$$

. which can be seen as the probability of measuring on a computational basis. For a probabilistic mixture of states, each with a probability as (pi, $|\psi_i\rangle$) the probability of measuring a basis state is calculated as

$$o_l = \sum_i p_i \, |\langle \omega_l | \psi_i \rangle|^2 \tag{5.8}$$

which can be approximated by as an average over M final states of several runs as

$$\hat{o}_l = \frac{1}{M} \sum_{j=1}^{M} \left| \langle \omega_l | \tilde{\psi}_j \rangle \right|^2 \tag{5.9}$$

The number of samples scales inversely quadratically in the desired precision ϵ in approximating the actual function. More details can be found in [151].

The stochastic quantum circuit simulation allows us to avoid the complexity of simulating $D \times D$ density matrices. However, the challenge remains to produce and process enough samples, or single states, which is still a complex computational challenge, as seen in Section 4.1.2. Using parallel processing could potentially reduce this constraint by conducting multiple simulations simultaneously on different processors; therefore, stochastic quantum simulation represents a balanced trade-off between optimizing memory and exploiting resources for concurrency by allocating individual cores to perform independent simulation runs. The Monte Carlo method for noisy quantum circuit simulation involves:

- Generating many instances of the circuit, each with noise applied stochastically.
- Simulating each instance.
- Averaging the results to obtain the expected behavior.

In the generation, the noise model logic determines whether a gate has a noise operation to be appended and, for entangling operations, whether noise is applied to the target, control(s), or a subset of qudits involved. Before this procedure, each gate is assigned a type of noise, whether qudit depolarizing noise or a particular type of subspace noise. If the gates operate in subspaces such as local subspace rotations [9], the noise matches the subspace of the operations. Hence, a noisy operation is sampled after each gate based on a success probability. The procedure is encapsulated in *NoisyCircuitFactory* within the pseudocode in the algorithm Algorithm 5. Figure 5.2 visually summarises the Monte-Carlo simulation, where parallel processes sample noisy circuits and then simulate them, obtaining a final state.

Algorithm 5 Monte Carlo Noisy Circuit Simulation

```
1: procedure NOISYSIMULATION(circuit, noise_model, num_shots)
2:     results ← []
3:     for i = 1 to num_shots do
4:         noisy_circuit ← NoisyCircuitFactory(circuit, noise_model)
5:         result ← execute(noisy_circuit)
6:         results.append(result)
7:     return results, AverageObservable(results)
```

Further information on the utilization of the implementation will be elaborated in *Part* IV, which introduces the open-source software for mixed-dimensional systems.

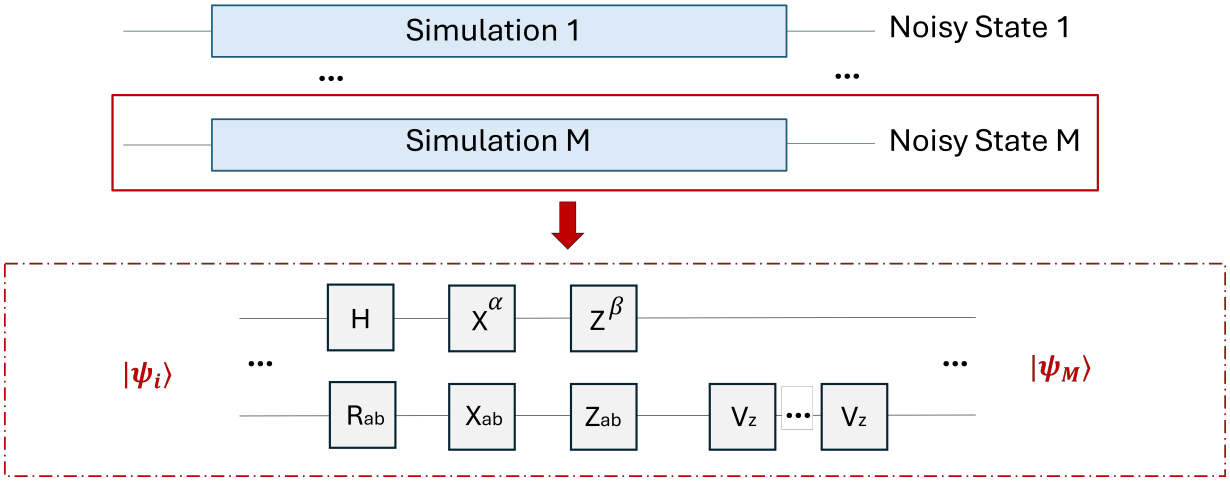


Figure 5.2 A Monte-Carlo quantum circuit simulation, based on the mixed-dimensional noise model.

Trading Numerical Correctness for Runtime in DDs

Using preexisting simulation engines can effectively leverage memory-efficient structures like DDs, even during concurrent operations. The size and complexity of decision diagrams are closely connected to the entries they represent. Considering that the numeric components in the matrix for a specific dimension of a qudit operation are part of a ring [154], [155] (a concept from number theory), this property impacts the complexity of decision diagrams, facilitating efficient constructions, quantum circuit synthesis [104], and simulation [156]. Managing complex entries within decision diagrams is a challenging topic [103]. If the efficiency in accessing memory and managing the accuracy of DDs depends on how the numerical entries are stored and retrieved from memory, then the design and application of simple subspace noise gates can significantly enhance the simulation performance of extensive yet shallow circuits. The simpler the ring of the operations, the more efficient the memory management of decision diagrams.

5.4 Conclusion

This chapter introduced novel approaches for simulating noisy quantum circuits, particularly emphasizing mixed-dimensional systems, making the first formalization of a realistic noise model for mixed-dimensional systems. Here, we presented a new specification of the depolarizing noise channel for both shallow and deep regimes and developed a Monte-Carlo framework using stochastic error modeling to calculate empirical averages across multiple simulation runs. Our methodology provides conceptual clarity and mathematical rigor while maintaining compatibility with existing tensor networks and decision diagram simulators. In future work, a particularly simple yet highly relevant extension is the integration of numerical distortions of the gate parameters in the circuit associated with a specific noise channel or the inclusion of subspace error operations for all types of gates. The details on how to use the implementation and the open-source code will be covered in *Part* IV.

6 Summary of Part II

This part of the thesis has explored the challenges concerning the classical simulation of quantum circuits, which is a relevant task for the correct development of quantum algorithm, and for the verification and validation of experimental results. Efficient data-structures and methods have been involved for solving or mitigating these challenges, like decision diagrams, as well as new models that could increase the usefulness of classical simulation in the tasks. In particular, this Ph.D. thesis presented the following contributions:

- *Chapter* 4 described a new method to facilitate the classical simulation of quantum circuits for mixed-dimensional qudit systems. This enhancement was achieved by introducing a novel type of decision diagram designed to manage the simulation of these circuits. Specifically, these decision diagrams encode the dimensionality of a qudit in the number of outgoing edges of a node. This efficient representation is at the base of the corresponding implementation, enabling an efficient simulation of quantum computations in practical scenarios. The experimental results validate its effectiveness for the chosen set of benchmarks, which range from simple circuits to random circuits that illustrate worst-case behavior.
- *Chapter* 5 presented innovative methods for simulating noisy mixed-dimensional quantum circuits. In fact, we introduced the first specification of the depolarizing noise channel that is applicable to both shallow and deep mixed-dimensional quantum circuits. In addition, we developed a Monte Carlo framework that combines stochastic error modeling to compute empirical averages over numerous simulation runs. Our approach offers both conceptual clarity and mathematical precision while remaining compatible with current tensor networks and decision diagram simulators.

The contributions become a baseline guide for verifying and validating quantum circuits in a realistic setting, understanding the physical limitations of an experiment, before even implementing an algorithm or a protocol on a quantum device, with flexibility and enhanced efficiency. All the contributions listed above have been developed in a push button solution, which is available open-source, and is described in *Part* IV, providing the quantum computing community with accessible resources to further advance the field.

Part III

Compilation of Quantum Circuits

7 Overview of Part III

The rapid growth of quantum computing, particularly with mixed-dimensional quantum architectures, which integrate both qubits and higher-dimensional qudits, requires the development of efficient compilation techniques. Similar to classical computing, where high-level languages like C++ enable the abstraction of complex low-level programming tasks, quantum computing needs compilers that can translate high-level quantum algorithms into executable low-level operations on physical hardware. The limited availability of automated compilation methods impedes the efficient utilization of quantum computing platforms for executing quantum algorithms.

Mixed-dimensional quantum computers are currently scaling in the number of qubits, qudits, and their dimensions. Implementing the quantum applications (such as those in Section 2.2.4) becomes a much more complex task. Quantum compilation plays an increasingly essential role in translating high-level algorithms into sequences of hardware-executable instructions that are optimized for resource usage and minimizing errors. In fact, maximizing a positive exploitation of the hardware capabilities requires constructing sequences that account for the faultiness of the operations, limited connectivity within and between qudits, and the variety of operations available in the gate-set (as discussed in Section 2.2.2). The ultimate goal is to execute computations on actual quantum hardware, and compilers for mixed-dimensional quantum computers aid in accelerating the design and development stages, optimize quantum circuits to reduce the inherent noise, increase the compatibility of the algorithms for different mixed-dimensional architectures, and improve the collaboration between the community of the algorithm and hardware designers.

Quantum compilers enable users to fully leverage quantum systems without the need to manually design circuits for each application, a process that would otherwise be time-consuming, error-prone, and poorly suited for adapting to new mixed-dimensional architectures or emerging technology platforms. Quantum compilation involves finding the shortest possible sequence of arbitrary quantum operations, in a procedure known as synthesis or decomposition, depending on the case, that can implement a given quantum state or operation as optimally as possible. The resulting quantum circuit must adhere to specific hardware constraints while minimizing error rates, often through a reduced gate count.

Efficient decompositions and synthesis methods also preserve properties such as entanglement structures and ensure the construction of states with high fidelity. However, compilation is severely constrained by memory demands due to the vastness of the search space, particularly in mixed-dimensional architectures where higher-dimensional qudits further increase state-space complexity. As a result, naive techniques quickly become unmanageable, even when substantial computational resources are available.

Part III of this Ph.D. thesis addresses several compilation tasks, each with its own set of challenges, and examines how the application of automated methods, also based on data-structures, like decision diagrams, graphs, quantum information, and heuristics, help in managing the memory and time complexity for constructing the sequences, or reducing the amount of resources needed. The methods in this thesis offer a systematic approach to the compilation of quantum circuits in mixed-dimensional quantum architectures, laying the groundwork for more efficient and scalable quantum software solutions. This part of the thesis presents four key contributions that systematically address various aspects of quantum compilation in mixed-dimensional architectures. Each of the following four chapters covers a specific aspect of our contributions.

Chapter 8, based on [11], explores a method for reducing the number of gates in qubit circuits, also called compression. The method identifies the optimal mixed-dimensional architecture for executing qubit-based programs, and understands how to transform the qubit circuit on the architecture. As a secondary result, the method performs resource estimation, providing valuable insights for tailoring mixed-dimensional architectures. *Chapter* 9, based on [12], examines an innovative synthesis approach for state preparation circuits that target arbitrary states within mixed-dimensional quantum systems. The method utilizes decision diagrams and optimization techniques for achieving the final quantum circuits in a versatile fashion. *Chapter* 10, based on [13], introduced a universal workflow for compiling any two-qudit unitaries into any hardware gate set that supports two-level entanglement gates. Avoinding manual intervention and system-specific implementations. *Chapter* 11, based on [14], presented an algorithm designed for the efficient decomposition of local unitaries within multi-level quantum systems. Traditional decomposition methods often produce sequences with significant redundancies; however, this new adaptable approach improves gate costs by considering realistic coupling constraints and rotation expenses, thanks to a new data-structure, the energy coupling graph.

These contributions represent the first systematic approach to compiling for mixed-dimensional quantum computers, establishing a foundation for future advancements in the field.

8 Compression of Qubit Circuits: Mapping to Mixed-Dimensional Quantum Systems

As discussed in Section 2.2.3, we must represent quantum algorithms as quantum circuits, which are sequences of quantum operations, to effectively control quantum computers. This sequence needs to be optimal in terms of the number, type, and ordering of operations to minimize the noise impact of each operation on the system, as also highlighted in *Chapter* 5. Various strategies have been suggested to improve these operations, such as by employing specialized heuristics (seen in Section 2.3.4). Shortening the sequences can also be considered a circuit compression, which can be achieved by tailoring a mixed-dimensional quantum architecture to the qubit circuit and realigning it accordingly. To identify the optimal mixed-dimensional architecture for running qubit circuits, it is essential to estimate the achievable compression or provide guidelines on the tailoring to maximize the compression.

This chapter, based on [11], introduces an efficient strategy to minimize the number of entangling operations, such as CX and CZ, in qubit circuits by mapping and transforming them into mixed-dimensional quantum circuits. We develop a method that models qubit interactions within a circuit using a graph-based approach. Using this graph, we applied a clustering technique over the qubit state space, enabling the grouping of qubits with recurring interactions. This process results in the specification of the target mixed-dimensional quantum architecture, including the dimensions of each qudit, which align closely with a user-specified preferred dimensionality. For the first time, this approach enables the compression of qubit-based quantum circuits into mixed-dimensional systems and provides tools for estimating resources and tailoring architectures to optimize performance. The evaluation demonstrates a significant decrease in the number of non-local gates to achieve a specific task, thereby reducing a major source of errors in quantum circuits. The implementation is open-sourced and shared with the contributions in *Part* IV.

The chapter is organized as follows: Section 8.1 exposes the problem considered, reviews state of the art, and illustrates the contributions of this paper. Section 8.2 elaborate on the proposed method for compressing qubit circuits into qudit circuits. Section 8.3 provides the results on evaluating the method. Finally, Section 8.4 concludes the chapter.

8.1 Motivation

In this section, we explore the concept of circuit compression, which is intended to optimize the resources needed for quantum computations. The specific use case focuses on minimizing the number of entangling operations by tailoring a mixed-dimensional quantum architecture and mapping a given qubit circuit onto it, enabling efficient utilization of the available resources.

8.1.1 Considered Problem

A method for assessing the cost of a qubit circuit involves counting the number of non-local gates—typically controlled phase rotations (CZ) or controlled negations (CX, also known as CNOT). This approach is driven by the tendency for error rates to increase as a result of more complex experimental controls as the number of involved qubits increases. Consequently, *circuit compression* seeks to enhance the computation's

quality by minimizing the number of operations within a sequence, with a particular emphasis on non-local operations and the number of qubits (or qudits) present in the circuit.

In more detail, circuit compression involves combining groups of qubits within a given circuit into qudits of appropriate dimensions and converting local and non-local operations within the circuit into local multi-level operations in the compressed circuit; the circuit can use qudits of variable sizes, making it mixed-dimensional. This method preserves the initial computation while optimizing the resource efficiency of the quantum hardware.

Example 17. *Figure 8.1 portrays the circuit compression procedure. The routine begins with a qubit (i.e., two-level) circuit consisting of numerous local operations and entangling operations among all the qubits. We can see that the first two qubits from the top frequently interact via two-qubit operations. Consequently, a* compressed *circuit (presented on the right-hand side) might merge these two qubits into a ququart (i.e., four-level qudit). This allows all operations originally performed on the first two qubits to be converted in local operations on the ququart. As a result, this circuit compression significantly reduces the number of non-local operations: only a single non-local operation is necessary between the ququart and the remaining qubit.*

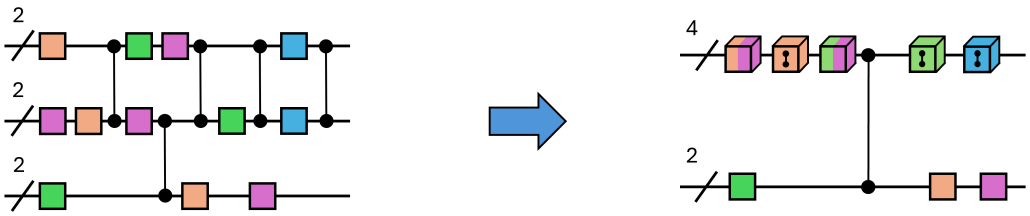


Figure 8.1 Illustration of a three-qubit circuit compressed into a ququart and a qubit. Squares denote qubit-specific operations, whereas cubes indicate multi-dimensional operations. Black lines connecting the black dots signify non-local operations(CZ). The compression process demonstrates that transitioning from a qubit-based circuit to a qudit-based one can minimize entangling operations, although at the expense of more complex local high-dimensional operations.

8.1.2 State of the Art

The subject of quantum circuit compression has gathered increasing attention among researchers recently. This area has led to insights into the conceptual relationship between qubit and qudit logic and the practical implementation of compressing qubit circuits. In the following discussion, we examine two recent studies demonstrating current circuit compression advances.

[56] introduces a proof-of-concept workflow aimed at compressing qubit circuits into a qudit platform. This workflow consists of three primary stages: transpilation, simulation, and post-processing. Importantly, this approach is contingent upon the assumption that qudit circuits are formulated using a specified set of high-level operations, including multi-controlled Toffoli gates.

The method outlined by [56] begins with the translation of qubit circuits onto a qudit platform characterized by a defined dimensionality for each qudit, thereby aiming to minimize qubits and entangling operations. This is achieved by enumerating all the potential mappings, a task feasible for smaller quantum circuits. The challenge of devising mapping strategies that can scale with the qubit circuit size in polynomial time remains unsolved. The linear-time complexity of transpiling non-local operations arises from the constraint of the study to a specific multi-qubit entangling gate set.

The work described in [157] discusses the challenge of compressing qubit circuits into qudit circuits while targeting qudits with a fixed dimension and focusing on multi-qubit entangling gates. The authors

present new bounds on circuit complexity after performing the compression, and they demonstrate that these bounds reveal a combinatorial complexity in the dimensions of the systems. The paper also reflects on the potential improvements in the expression of non-local operations on the qudit space, with concrete experimental implementations across various technologies. The introduction of algorithmic strategies and heuristics to establish an efficient mapping for quantum circuit compression, guided by the physical constraints of the qudit platform, remains an unresolved issue.

The study presented in [158] introduces a compression strategy designed for superconducting ququarts, which involves the development of a custom set of gates crafted to execute qubit-ququart operations. Additionally, the authors introduced a corresponding mapping method to allocate qubits to ququarts. Despite the completeness of the approach, it is highly specific to the chosen technology and restricts the number of levels to a maximum of four. Still various technological platforms have the potential to surpass the performance of the targeted architecture examined.

8.1.3 Contribution

This chapter investigates quantum circuit compression from arbitrary qubit circuits into mixed-dimensional quantum systems to enable potential benefits in terms of improved circuit efficiency. For this reason, a new design automation tool that allows for automatic and efficient quantum circuit compression is presented.

The proposed quantum circuit compression approach, which maps qubit circuits to qudit circuits, consists of two phases:

1. Encoding the qubit logic into a Hilbert space generated by the combination of suitable higher dimensional carriers and

2. translating the non-local qubit operations into either
    - corresponding non-local operations between mixed-dimensional qudits, or
    - local operations within a qudit.

In the first phase, we can implement the encoding by forming the tensor product of individual qudit state spaces. A critical aspect of this phase involves determining optimal dimensions for each qudit. Our novel algorithm groups qubits within the original circuit to minimize inter-cluster non-local operations, which will later be consolidated into qudits. The mapping method improves the performance of the compilation algorithm by establishing an optimized logical encoding, resulting in more compact circuits than conventional approaches such as ladder-coupling schemes [13]. Additionally, the mixed dimensionality avoids wasting efforts in controlling information not necessary for the computation.

The second phase covers the translation of the non-local qubit operations into genuine entangling two-qudit operations. This is important to ensure that the resulting qudit circuit can be executed on a physical quantum computer with the appropriate non-local operations in the higher dimensions. More in particular, while the mapping routine will be a necessary and useful preliminary step, a dedicated compiler for high- and mixed-dimensional quantum systems is a suitable candidate to produce close-to-optimal results. The simple translation of the qubit entangling operations to qudit ones would require the construction of correct pulses by optimization. However, the strength of mixed-dimensional systems is the richer entangling gate set given by the choice of a specific technology.

An example illustrates the idea behind both the proposed steps.

Example 18. *Figure 8.2 illustrates the mapping from a four-qubit circuit with a single three-controlled Toffoli (MCX) gate. Implementing the MCX is rather expensive in terms of the number of operations. First, the original gate is transpiled into single-qubit gates and CZ gates; the physical interactions between the qubits*

are observable and can be used to cluster the qubits. In this case, the clusters are on the first two and last two qubits, which subsequently get compressed into two ququarts (qudits of dimension four). The translation step then results in only one gate, the controlled-exchange [9] (CEX*) gate.*

In the next section, we introduce an algorithm that efficiently compresses qubit circuits into qudit circuits. The proposed solution will, for the first time, enable circuit compression for mixed-dimensional systems.

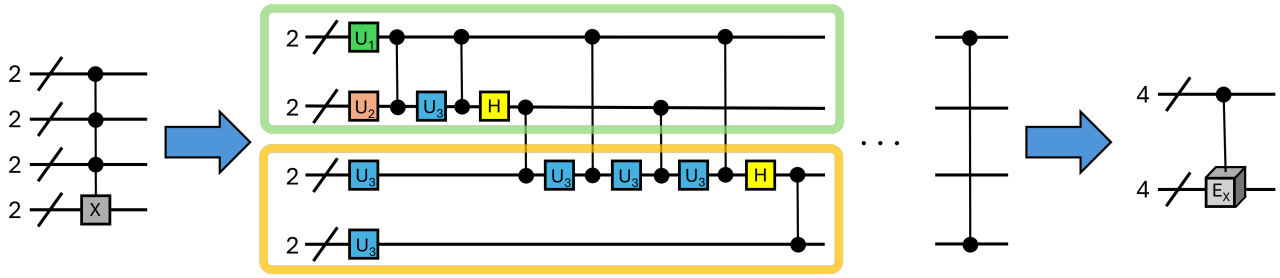


Figure 8.2 On the left the multi-controlled Toffoli, MCX gate. On the right, a qubit circuit representing the transpiled version of the MCX gate to two-qubit interactions and local gates (U_1, U_2, U_3 and Hadamard). The two colored outlines represent the individual qudits, where sub-portions of the qubit circuit are assigned to, depending on the results of the mapping [11].

8.2 Mapping to Mixed-Dimensional Systems

This section presents the methodology for compressing qudit circuits into circuits with mixed-dimensional qudits. More in particular, we outline our approach for clustering qubits in a given circuit based on their local and non-local operations, then converting these clusters into a circuit using qudits with corresponding dimensions. However, first, we examine the advantages of various qudit platforms and entanglement structures, as these factors are crucial in determining suitable mappings.

8.2.1 Rationale for Qudit Systems

Qudit technology platforms present several challenges but promise to show near-term advantages over equivalent qubit realizations. In fact, qudit systems have lower decoherence rates [16], they can use higher energy states to implement error correcting codes [159] and more efficient encoding schemes [160].

The comparative study in [161] compares all the most prominent qudit technologies in terms of gate efficiency for systems composed of sets of qubits and qudits and allows for an analysis of the trade-offs required between scaling the information density inside a quantum circuit and managing noise error rates. The efficiency of qudit gates must always be larger than the multi-qubit one by a factor $\mathcal{O}(d^2/\log_2(d))$. Given Hilbert space of equivalent dimensions, viable qudit platforms are capable of outperforming equivalent state-of-the-art multi-qubit ones in gate fidelity for pure dephasing. Moreover, this performance could be extended to qudits with d as large as 40 or more. This analysis naturally leads us to reflect that we can perform qubit circuit compression with different performances depending on the circuit structure, but especially on how well the chosen technology platform can implement d-leveled circuits. To achieve the most efficient compression of a specific qubit circuit, we must choose the most suitable dimensionality for the target qudits and consequently select the right technology that can efficiently implement that circuit.

8.2.2 Entanglement Structures

Entanglement is a powerful quantum mechanical effect and a key component in the efficiency of quantum computations. Two or more qubits are in an *entangled* state, if their state cannot be written as a product of states of the individual qubits. Moreover, the *entangling power* or *structure* of non-local operations corresponds to the amount of entanglement generated by an operation, accompanied by the analysis of how the constituent basis states of an entangled state are affected by it.

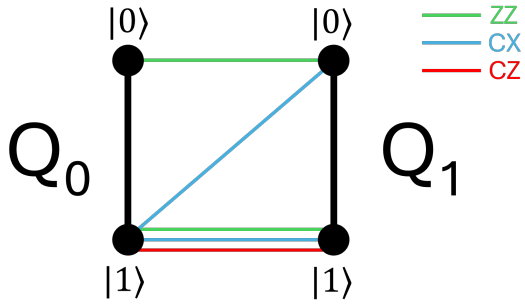


Figure 8.3 The black solid lines are local operations between basis states, the black dots. The colored lines are the entangling structure of CZ, CX, and CZ on two qubits.

Example 19. *Figure 8.3 illustrates three different examples of entangling structures for two qubits.*

- ZZ *(green line) entangles* $|0\rangle$ *and* $|1\rangle$ *of* Q_0 *with* $|0\rangle$ *and* $|1\rangle$ *of* Q_1*, respectively.*
- CX *(red line) entangles* $|1\rangle$ *of* Q_0 *with* $|0\rangle$ *and* $|1\rangle$ *of* Q_1*.*
- CZ *(red line) entangles* $|1\rangle$ *of* Q_0 *with* $|1\rangle$ *of* Q_1*.*

The graph is derived from the matrix representations of the operations and provides an insight on the physical application of the pulses representing these operations. The three operations present three completely different entangling structures; although the CX and CZ have the same amount of entropy or entanglement, these physical measures are spread in different ways, as graphically represented. The ZZ has double the amount of entanglement, and we can visually represent it as the application of two CZ on the upper and lower nodes of the graph.

8.2.3 Mapping Qubit Circuits by Clustering Weighted Graphs

As previously discussed, the purpose of constructing a mapping from qubit circuits to mixed-dimensional quantum systems involving appropriate qudits is to minimize both the number of qudits and the number of non-local operations. In this section, graphs are used to represent quantum circuits, where each node matches a level in the qubit state, and the edges correspond to non-local operations when the two nodes belong to two different qubits or local operations if the two nodes belong to the same qubit.

Due to the increased relevance of the the single levels of the qubit, the representation of each qubit is split in the two basis states 0 and level 1, as in Figure 8.4. In the graph, many local or non-local operations between the corresponding nodes are signified by edges with a high weight. These nodes are considered good candidates to be included in a cluster.

To convert a specific circuit into its graph representation, it is initially transpiled to consist exclusively of unitary local operations and controlled-Z (CZ) gates. This allows for clear distinction between local and non-local operations, especially when complex custom gates are employed. The use of the CZ gate simplifies the interactions in the final graph since it involves only a single-edge link between the $|1\rangle$ states of

two separate qubits (like in Figure 8.3). Additionally, this gate is well-researched and possesses beneficial properties for use with two qudits [157], [162].

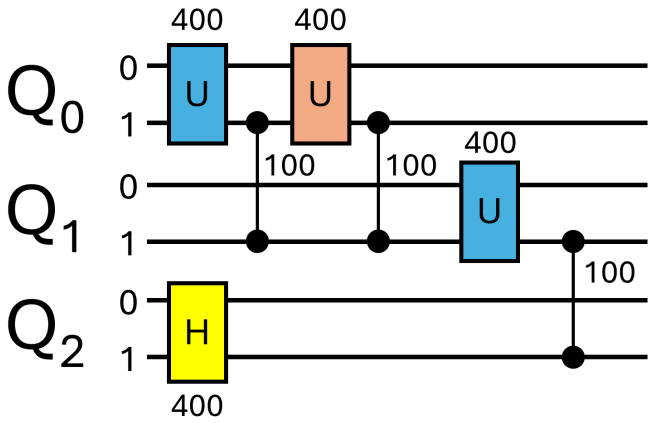


Figure 8.4 A transpiled circuit where the operations are applied to the single levels, the colored gates are respectively U gates and Hadamard. There are weights assigned to each operation.

Each operation within the quantum circuit is associated with a weight, and the edge corresponding to an interaction maintains the cumulative sum of the weights from all operations implemented on it. The weight symbolizes the attraction between two separate levels in the graph. Moreover, the weight of local operations is 4 times greater than that of CZ gates.

Although the error rates of entangling operations are an order of magnitude higher compared to local ones, the implementation cost for a qubit local operation, when transpiled into a non-local operation within a qudit circuit, is approximately estimated to be at least four times greater. This estimation presumes that a local operation, when converted to an entangling operation, will necessitate multiple swap operations to be feasible. In this manner, the graph also permits the division of the logic of a single qubit across two qudits, when needed. This becomes beneficial when a qubit lacks local operations yet is often employed as a control or ancilla. The control level, alongside the target qubit, can be mapped within the same qudit, thus providing a local connotation to the previously entangling relation.

Once the weighted graph is completely calculated, levels with high interaction are ideally mapped to the same qudit. This is achieved by using the K-means algorithm. We refer to the GCLU software that implements the K-means and M-algorithm for graph clustering, [163] presents more details on the algorithms.

The algorithm is designed to efficiently categorize nodes so that the total weight of the internal edges within each cluster is maximized while also reducing the total weight of the edges that connect different clusters. Because mapping to qudit systems requires minimizing entangling operations, the algorithm consequently reduces the number of qubit operations and entangling operations between qudits that need to be transpiled.

Running K-means on the graph typically results in configurations where the final qudits involve only local operations. To address this limitation, we propose an alternative construction that accounts for qudits without local operations, which we called *full connectivity*. In this approach, the initial graph construction is the same, but an additional step is introduced: every pair of nodes lacking a direct edge is connected by an edge with a minimal non-zero weight. Figure 8.5 illustrates the effects of clustering a graph based only on the circuit's connectivity.

Figure 8.6 shows how once the final mapping is produced, the transpilation step can take place. For this step, the efficient realization of the entangling operations between qudits is assumed to be solved externally from the tool by a compiler, and that the experimental control is up to the task [11].

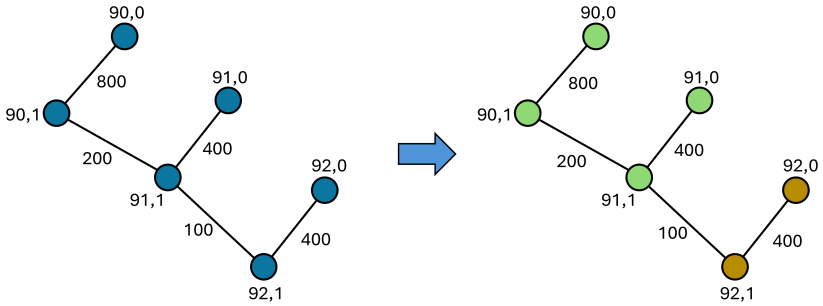


Figure 8.5 Initial and clustered graph. The left-hand side of shows the corresponding initial (unlabeled) graph of levels and weights between the levels. The right-hand side shows the clustered graph with a ququart (green nodes, four dimensions) and a qubit (red nodes, two dimensions).

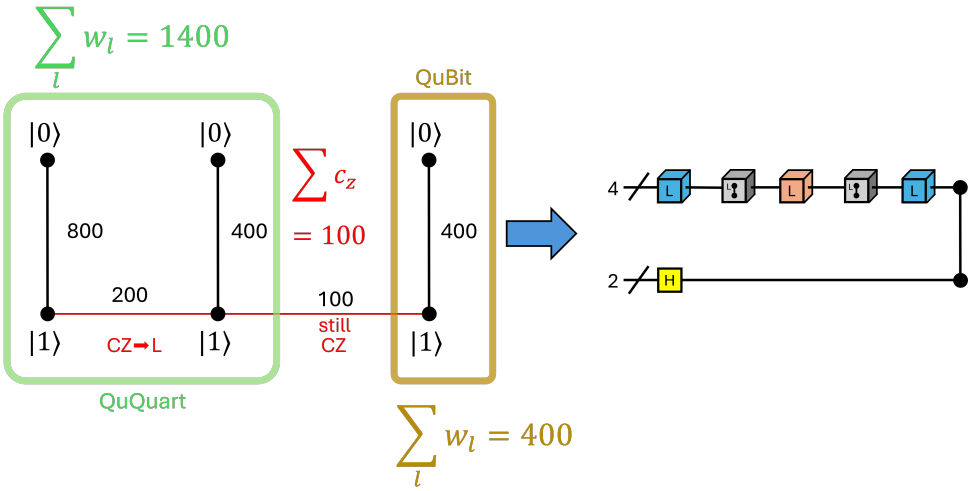


Figure 8.6 Compiling into qudit circuit–illustrates a clustering on a graph as well as the correspondingly compiled circuit.

8.3 Evaluation

In this section, we evaluate the method introduced in this chapter. For this reason, we examine a group of established algorithms operating on qubits and transform them into mixed-dimensional circuits, circuits with qudits of different dimensions once again. The implementation is entirely in Python 3, except for external dependencies such as Qiskit [3] and GCLU Software [163]. This section offers benchmarks for compressing circuits into mixed-dimensional systems, achieved through the development of a novel mapping strategy for qubit circuits towards high and mixed-dimensional systems.

The evaluations were conducted on a server operating GNU/Linux with an Intel(R) Xeon(R) W-1370P processor (at 3.6 GHz) and equipped with 128 GiB of main memory. The execution time depends on the version of Qiskit [3] and Python used for initial parsing and analysis of the qubit circuit, as well as the number of iterations of the clustering algorithm executed. The clustering algorithm is executed 20 000 times, with 4 preferred as the cluster dimensions. The clustered physical dimensions are translated into individual qudit spaces' dimensions. For each benchmark considered, the total execution time is several seconds.

The results are presented in Table 9.1 Here follows the discussion of the results based on the original work [11]. The first group of columns contains the basic information on the considered benchmark: The name of the algorithm, the number of qubits, and the number of CZ gates after decomposition. The following

two column groups contain the information on the resulting qudit circuits for “Circuit Connectivity” (edges in the graph correspond to the connections of the states according to the circuit) and “Full Connectivity” (all nodes in the graph are pair-wise connected). The “Qudit” column lists the dimensions of the qudits in the resulting circuit in the form *Dimension x Count*, e.g., 4x3 denoting three qudits of dimension four. Following, the columns “#CZ” give the number of non-local CZ operations in the circuits—with the “Ratio” between the compressed circuit and the initial qubit circuit. In many of the benchmark instances, the compressed circuits need less than 50 % of the non-local gates compared to the initial qubit circuit. The method has always success even if with a small percentage of improvement.

There is an instance of the Max-Cut problem, solved with two different algorithms, namely QAOA and VQE. One set of solutions is squarely set around 40 % to 50 %, while the other group is around 90 %. Circuit structure plays a major role in mapping to a mixed-dimensional architecture, albeit the study of the reasons is beyond the scope of this chapter.

The results confirm a reduction in the number of non-local operations when mixed-dimensional qudits are used to realize the circuit. This reduction directly translates to a decrease in one of the primary contributors to errors in quantum operations for these circuits. Furthermore, the tool effectively estimates the compression of qubit quantum circuits on mixed-dimensional quantum architectures and serves as a recommendation tool for tailoring the architecture to achieve optimal compression.

Table 8.1 Evaluation of the compression approach comparing the number of required non-local CZ gates

Benchmark			Full Connectivity			Circuit Connectivity		
Name	Qubits	#CZ	Qudits	#CZ	Ratio	Qudits	#CZ	Ratio
Amplitude Est.	10	90	[8x1,2x1,4x3]	78	0.867	[8x1,2x1,4x3]	78	0.867
	31	918	[8x2,16x1,2x5,4x8]	878	0.956	[128x1,2x6,4x9]	858	0.935
GHZ State	10	9	[4x5]	4	0.444	[4x5]	4	0.444
	15	14	[2x1,4x7]	7	0.500	[2x1,4x7]	7	0.500
	31	30	[2x1,4x15]	15	0.500	[8x1,2x2,4x13]	15	0.500
Graph State	10	10	[8x1,2x1,4x3]	4	0.400	[16x1,1x1,4x3]	6	0.600
	15	15	[2x1,4x7]	8	0.533	[2x1,4x7]	8	0.533
	31	31	[2x1,4x15]	16	0.516	[2x1,4x15]	16	0.516
Grover no-Ancillas	10	39 032	[32x1,2x3,4x1]	36 924	0.946	[32x1,2x3,4x1]	36 924	0.946
QAOA (Max-Cut)	10	40	[8x1,2x1,4x3]	16	0.400	[8x1,2x1,4x3]	16	0.400
	15	60	[2x1,4x7]	32	0.533	[2x1,4x7]	32	0.533
Quantum Phase Est. Inexact	10	102	[8x1,2x1,4x3]	81	0.794	[8x1,2x1,4x3]	81	0.794
	15	231	[16x1,2x3,4x4]	202	0.874	[16x1,2x3,4x4]	202	0.874
	31	963	[64x1,2x5,4x10]	886	0.920	[32x1,2x4,4x11]	891	0.925
VQE (Max-Cut with TwoLocal ansatz)	10	90	[4x5]	80	0.889	[4x5]	80	0.889
	15	210	[2x1,4x7]	196	0.933	[2x1,4x7]	196	0.933
W-State	10	18	[4x5]	8	0.444	[4x5]	8	0.444
	15	28	[2x1,4x7]	14	0.500	[2x1,4x7]	14	0.500
	31	60	[2x1,4x15]	30	0.500	[2x1,4x15]	30	0.500

“Ratio” is the fraction of CZ gates in the compressed and initial circuits.

8.4 Conclusion

In this chapter, based on [11], we introduced a method to improve the quality of the runtime of qubit quantum circuits. Although qudits offer a richer state space, most current quantum algorithms are designed for qubits. Our proposed method allows one to leverage the advantages of both approaches: algorithms can be structured easily around two-level qubits, while the hardware can efficiently exploit higher-dimensional qudits during execution. The results of our evaluations demonstrated a significant reduction in the number of

non-local quantum operations, which directly translates to a decrease in one of the primary sources of errors in quantum computation. Furthermore, the developed tool estimates the compression of qubit quantum circuits on mixed-dimensional quantum architectures and acts as a recommendation tool for tailoring the architecture to achieve optimal compression. The open-source software contributions discussed in this chapter form a part of what will be presented in *Part* IV. Potential future work includes deeper integration of hardware-specific considerations to form clusters that align more closely with the targeted qudit platform's gate sets and constraints, further enhancing the method's applicability and performance.

9 Mixed-Dimensional Qudit State Preparation Using Edge-Weighted Decision Diagrams

With the advent of robust mixed-dimensional quantum devices and novel quantum algorithms, challenges commonly encountered in qubit systems are expected to arise. In this context, methods for state preparation become increasingly important, particularly for applications such as those discussed in Section 2.2.4, including quantum simulation [64], [74], [76] and quantum machine learning [164]–[166]. These applications rely heavily on the accurate preparation of specific initial states to function effectively. While Section 2.2.2 and Section 2.2.1 provided an overview of quantum states and their manipulation, many aspects of these states, such as entanglement, remain poorly understood in qudit systems. Studying state preparation is critical for advancing our understanding of quantum states and unlocking the full potential of various quantum technologies. Hence, addressing the challenges of state preparation is essential not only for improving quantum simulations but also for deepening our understanding of quantum information science.

This chapter, based on [12], introduces a novel method for synthesizing state preparation circuits, aimed at reproducing mixed-dimensional quantum states using minimal operations and controls, which are then transpiled for specific architecture gate-sets. For this reason, in this chapter we present three contributions:

- A new use of edge-weighted decision diagrams for a compact and efficient representation of *mixed-dimensional* states.
- An automated procedure to generate quantum circuits for the construction of arbitrary quantum states.
- Moreover, a new implementation of approximation methods for decision diagrams has been examined to enhance the circuits realized.

We illustrate the benefits and capabilities of the proposed method by compiling a diverse range of uniform and random quantum states. These states are compiled into quantum circuits using qudits with mixed-dimensions. This is the first instance of demonstrating the automatic and efficient creation of mixed-dimensional quantum circuits capable of constructing arbitrary states. Moreover, these circuits can be further optimized by integrating approximations without significantly reducing their fidelity. The open-source implementation is integrated into the contributions explained in *Part* IV.

The remainder of this chapter is organized as follows: Section 9.1 describes the problem considered, reviews the state of the art, and summarizes the contribution of this paper. Section 9.2 presents the proposed approach for synthesizing quantum circuits for qubit-qudit circuits in detail. Section 9.3 discusses the evaluation of the implementation of the proposed approach. Finally, Section 9.4 concludes the chapter.

9.1 Motivation

We examine the problem of state preparation in quantum circuits with mixed dimensions. After discussing previous research in this area, we present our main contribution: a new approach to scalable state preparation using decision diagrams adapted for mixed-dimensional systems.

9.1.1 Considered Problem

Quantum state preparation is a topic of theoretical and practical relevance. In fact, it plays a crucial role and determines the efficiency of inputting classical data into a quantum computer. Furthermore, it serves as a critical subroutine for various quantum algorithms, including well-known ones [50], [51], [167], as well as quantum machine learning [164], [168], [169], and Hamiltonian simulations [165], [166]. Formally, quantum state preparation involves preparing a given quantum state $|\psi\rangle$ from an initial product state $|0\rangle^{\otimes N}$, using single and two-qudit gates of different types, as shown in Figure 9.1. However, quantum operations are prone to errors due to factors like limited coherence and gate infidelity, which reduce the accuracy of quantum computations. These challenges can be addressed through methods that minimize the number of operations while maintaining reliable results.

Hence, the state preparation problem is to find the shortest sequence of quantum operations that can implement a state $|\psi\rangle$, starting from the product state $|0\rangle^{\otimes N}$, while achieving a desired quantum state fidelity, in the most computationally efficient fashion.

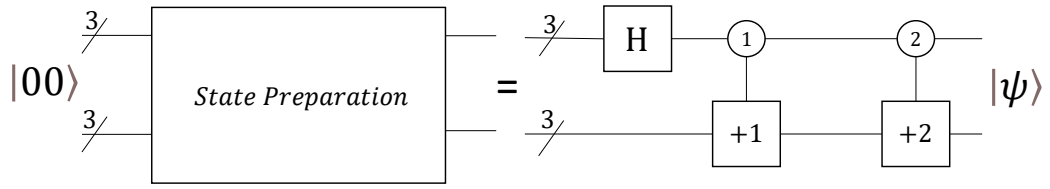


Figure 9.1 The state preparation algorithm for constructing $|\psi\rangle$, compiled into a sequence of single and two-qudit operations, in this case a two-qutrit GHZ state.

Example 20. *In this example, taken from [12], let's consider the realization of the quantum state* $\frac{1}{\sqrt{3}}(|00\rangle + |11\rangle + |22\rangle)$*, i.e., a GHZ state, in a qutrit-qutrit system. The state is entangled and a superposition of three basis states* $|00\rangle, |11\rangle, |22\rangle$*. The problem is to find a sequence of operations that can transform the state* $|00\rangle$ *into this state. Figure 9.1 shows a possible result: the qutrit Hadamard generates the superposition as shown in Example 5. Then, controlled operations on the state of the first qutrit manipulate the second one. Each operation is controlled by the level inside the circle and imposes an increment based by "+1" or "+2".*

9.1.2 Related Work

Earlier techniques have been suggested for the preparation of arbitrary quantum states, although these methods have mainly concentrated on qubit quantum circuits and have been customized for specific applications or particular classes of states [170]–[172]. More recent progress in circuit depth and complexity of state preparation circuits has been proved possible, although this comes at the cost of requiring an exponential number of auxiliary qubits [173]. Decision diagrams have also proved useful in the context of state preparation for specific classes of states, such as uniform states, cyclic states, and arbitrary states, although the methods required ancillary qubits [171], [172]. The success of these methods show the effectiveness of decision diagrams in automating state preparation. Furthermore, experimental designs have been developed and conducted for a direct construction of the states [174]. While scalable algorithms have been presented for the case of prime-dimensional qudits w-states, or their qubit-embedded version [147], the investigation and development of scalable automated procedures, particularly optimized ones, for qudit circuits and mixed-dimensional systems, remain unexplored.

9.1.3 Contribution

In this chapter, we examine state preparation methods for mixed-dimensional quantum systems, which enable various quantum algorithms on such architectures while automating complex routines involved in quantum computations. The proposed approach aims to improve the circuit efficiency in a scalable manner. To achieve this, we introduce a tool that enables automatic and efficient quantum state preparation using decision diagrams for mixed-dimensional systems.

We present three contributions that realize the method:

- The primary contribution is the study of the state preparation problem using a newly developed data structure known as the edge-weighted decision diagram with a variable number of successors. This data structure allows for a more precise and realistic description of quantum systems, supporting both qubits and qudits of various dimensions similar to the inherent diversity encountered in complex quantum simulations. This approach is contrast with traditional decision diagrams [98], which were limited to processing uniformly sized units of information. Additionally, this research is particularly important as there is no previous literature addressing state preparation using decision diagrams with a variable number of successors.

- The second contribution of this work focuses on the synthesis of high-level quantum circuits, which enables the automated state preparation procedure in an efficient manner. The routine's complexity is linear in the number of nodes in the decision diagram. This significant development is pivotal in creating states that would otherwise be challenging and time-consuming to design manually. Automating the state preparation process enables researchers to manage intricate quantum states efficiently, opening up new opportunities and expediting advancements in quantum information processing.

- The last contribution involves the application of approximation techniques, with the goal of decreasing the gate count of the quantum circuits generated. Particularly, these techniques aim at the elimination of operations that generate portions of the state that are practically irrelevant for the computation, by leveraging the reduction in the size of the decision diagram representing the desired state. While previous work has focused on this aspect for *qubit* circuit simulations [175], the present study expands and generalizes these routines to qudit systems, extending their utility beyond their original purpose. Moreover, new reduction rules have been introduced to simplify decision diagrams. These approximation techniques simplify the final logic, and by removing nodes from the decision diagram, they reduce the complexity of the resulting quantum circuit. This reduction identifies patterns corresponding to tensor operations between subspaces of single qudit Hilbert spaces, which manifest as DD subgraphs. These complex patterns, previously unanalyzed, are computationally challenging. An additional benefit is the reduction in required control operations, enabling more resource-efficient operation sequences.

9.2 State Preparation

In this section, we propose the synthesis method that converts state decision diagrams into mixed-dimensional qudit circuits. To this end, we begin by introducing decision diagrams, which efficiently store multi-dimensional quantum states. Afterwards, we illustrate how these decision diagrams are used to create quantum circuits that generate the desired states. Lastly, we examine how approximation techniques can further optimize the resulting circuits.

9.2.1 Decision Diagrams

Previously, decision diagrams have been shown to facilitate efficient representations of exponentially large data in numerous instances [98], [100], [101], [130], owing to their ability to achieve compactness by capitalizing on the redundancy present in the data they depict. Before getting to the details of the method, we suggest to revisit the background on the data structure, which is explained in detail in Section 2.3.1 and Section 4.2.

This following procedure describes the first contribution and part of the procedure corresponding to the first arrow in Figure 9.2, in other words, the representation of a quantum state as a decision diagram, as described in [12]. Consider an n-qudit quantum state defined over a set of variables $q_{n-1}, q_{n-2}, \ldots, q_0$, where q_{n-1} w.l.o.g. is assigned as the "most sigificant qudit". Firstly, the state is divided into parts of equal size, based on the dimensionality of qudit q_{n-1}. Each part is associated with a successor node, which also contains the decision regarding the value of q_{n-1}. This splitting process continues for each sub-vector until the individual complex entries in the original vector are obtained. Throughout this process, the complex amplitudes of each basis state are stored in the form of weights inside the edges on the path from the root to the leaf, followed during the decomposition. To ensure consistency, the nodes are normalized such that the sum of the squared magnitudes of the out-edges from a node adds up to one. To reconstruct the amplitude for specific basis states, one can traverse the decision diagram accordingly, multiplying all edge weights along the chosen path.

The resulting decision diagram consists of a *root node* q_{n-1}, at least one node for each subsequent q_i, and ultimately a single *terminal node* without any successors. An example illustrates the procedure.

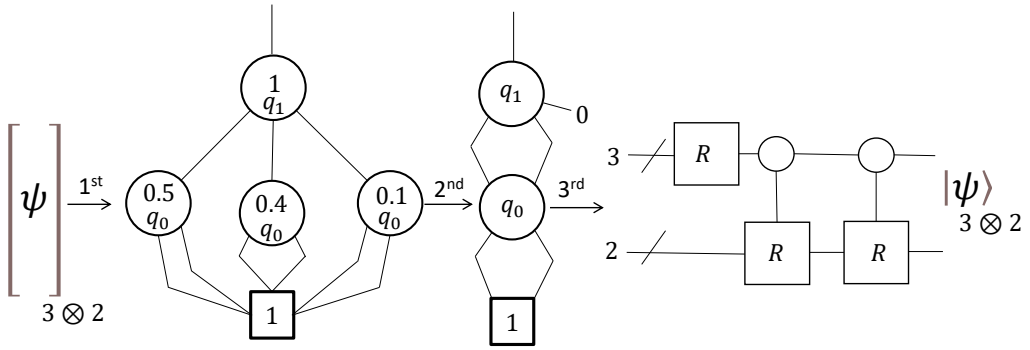


Figure 9.2 The three steps of state preparation.

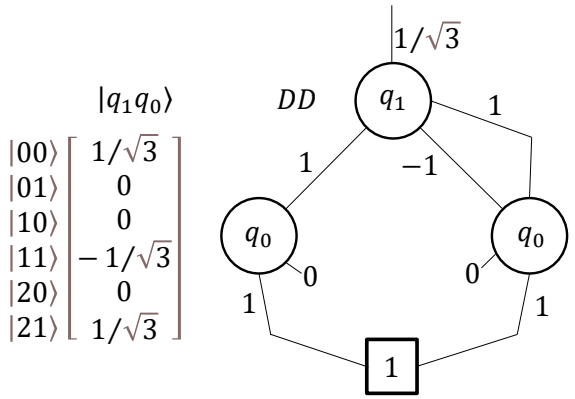


Figure 9.3 A state vector of circuit composed of a qutrit and a qubit and the corresponding decision diagram representation.

Example 21. *Here an example of the construction of the DD starting from a state, inspired by the original work in [12]. Consider the quantum state* $\frac{1}{\sqrt{3}}(|00\rangle - |11\rangle + |21\rangle)$ *in a qutrit-qubit system, represented in 2 forms in Figure 9.3. The vector's dimension is 6, which results from combining the local dimensionalities of the qutrit* 3 *and the qubit* 2*. The root node, denoted as* q_1*, branches into 3 edges, each corresponding to a distinct level in the qutrit. Moving to the* 2^{nd} *level, we encounter nodes representing the qubit, with each node having 2 edges. Notably, the 2nd and 3rd edges of the root node connect to the same qubit node, making use of redundancy. The nodes labeled as* q_0 *direct to the* terminal node*. To calculate the amplitude of a specific basis state, we multiply the weights along the path corresponding to that basis state. For instance, for the bitstring* $|11\rangle$*, the computation involves multiplying* $1/\sqrt{3} \cdot -1 \cdot 1$*, which accounts for the weights of the root node, the 1st edge of* q_1*, and the 1st edge of* q_0*.*

9.2.2 Synthesis Algorithm

The algorithm begins by recursively building a decision diagram that depicts the quantum state, considering the dimensionalities of both qudits and qubits, as outlined in the prior section. Due to the differences in dimensionalities among the information units, nodes at the same level may have an equal number of successors, whereas nodes at different levels might show varying dimensionalities. Each link between a parent node and its successor is given a complex number weight, which is the normalizing factor derived recursively from the successor's outgoing edges. This normalization process starts at the edges of the terminal nodes. The weights are normalized according to a predetermined scheme to maintain node canonicity and achieve a more compact representation following the reduction stage. For the normalization of a node's out-edge weights, each weight is divided by the normalization factor, ensuring that the squared magnitudes of all edge weights sum to one. This norm is subsequently applied to all in-edge weights of the node in question. At this point, the decision diagram forms a weighted tree and is fully prepared for the traversal that will produce the quantum operations for constructing the desired quantum state.

Now we proceed by describing the generation of the quantum circuit beginning from the decision diagram, the contribution of this work corresponding to the third arrow in Figure 9.2. The routine is implemented recursively, beginning with the root node and traverses the decision diagram. It processes the successor edges, starting from the last one, in pairs of two and in descending order. At each iteration, the algorithm executes a two-level rotation, referred to as a Givens rotation [9], on the two adjacent edges or dimensions signified by the node. The rotations between two levels i and j are expressed as

$$R_{i,j}(\theta, \phi) = \exp\left(\frac{-i\theta}{2}\Big(\cos(\phi)\sigma_x^{i,j} + \sin(\phi)\sigma_y^{i,j}\Big)\right),$$

where $\sigma_{\{x,y\}}$ are the Pauli-$\{X, Y\}$ matrices, θ is the rotation angle, and ϕ is the phase of the rotation. These parameters are calculated as: $\theta = 2 \cdot \arctan[|w_i/w_j|]$, $\phi = -(\pi/2 + \arg[w_j] - \arg[w_i])$. The sequence finishes with a phase rotation applied on the level 0-1, but with angle as the phase difference between the level 0, calculated from the sequence, and the father's node weight. The rotation can be decomposed into two-level rotations using the identity $\mathrm{Z}(\theta) = R(\frac{-\pi}{2}, 0) \cdot R(\theta, \frac{\pi}{2}) \cdot R(\frac{\pi}{2}, 0)$.

Every rotation will be applied to the circuit with a set of qudit controls equivalent to the set of nodes encountered on the path from the root to the node of the edges considered. For each node, the control level of the operation will be the index of the edge taken to descend the decision diagram.

Example 22. *Figure 9.4 shows a step of the synthesis algorithm. The algorithm appends to the circuit the rotation R on levels 1 and 2, and calculates the parameters accordingly. The rotation is controlled on level 1 of the the first qutrit since the rotation was derive from the node with index 1.*

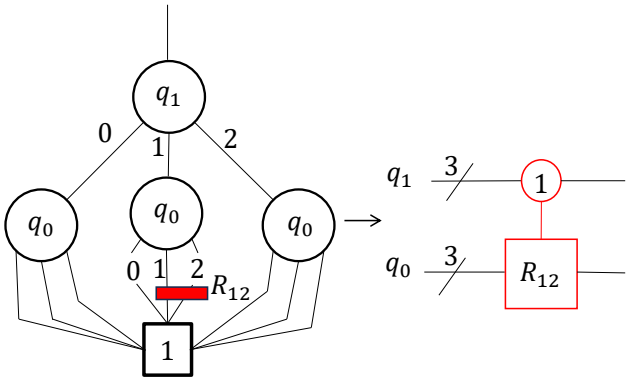


Figure 9.4 Example of a DD representing the state of a circuit composed of two qutrits and of a rotation synthesized from it.

9.2.3 Optimization and Approximation

In this section, we discuss a segment of the contribution represented by the second arrow in Figure 9.2. We utilized various optimizations in the decision diagram construction phase to enhance the synthesis of more compact circuits. The initial optimization is derived from research in quantum circuit simulation. Approximated decision diagrams proved an effective solution for improving memory complexity and runtime performance [175]. Nevertheless, the method comes with a compromise in state fidelity, requiring a careful equilibrium between computational efficiency and the precision of outcomes. The technique is a generalization of [175], where it calculates the contributions in terms of the fidelity of each node and then removes the nodes from the decision diagram until a threshold fidelity, previously chosen for the synthesis, is reached. The contribution is calculated as the sum of the squared magnitude of each amplitude that has a path from root to leaves crossing the node.

The benefits of the technique are the following, the method:

- reduces the size of the decision diagram in terms of memory.
- Reduces the synthesis time, since its complexity is tied to the size of the decision diagram.
- Lastly, the technique enables synthesizing shorter circuits for the state preparation, with guarantees on the fidelity reached by the procedure.

The second optimization consists of reducing the decision diagram, intended as the capability of two edges pointing to the same node, whenever it represents two identical sub-trees, that would be otherwise stored twice. The advantage here lies in the logical significance of this reduction. When all edges with weights other than zero point to the same node, it resembles a tensor product operation between the two qudits representing adjacent levels in the tree. Consequently, operations in the sub-tree will not consider the father node with successors pointing to the same child as a control qudit, thereby reducing the number of entangling gates during transpilation.

Example 23. *Figure 9.2 illustrates the implementation of optimization techniques. The successor node of the root, contributing the least to fidelity (0.1), is pruned from the decision diagram. The two residual edges connecting to nodes representing identical sub-vectors are redirected to a common node. This approach reduces the memory needed for node storage and, owing to the characteristics of tensor products, eliminates the need for control synthesis.*

Overall, the method results in a state preparation procedure of three main steps, as depicted in Figure 9.2: starting with a representation of the state as a decision diagram, followed by an approximation routine to reduce the size of the DD, and concluding with the realization of optimized quantum circuits.

9.3 Experimental Evaluation

To evaluate the feasibility of the proposed quantum state preparation approach, we implemented the method and evaluated it on several types of quantum states. The selected benchmarks are

- Embedded W-State [147],
- GHZ State [148],
- W-State [176], and
- Random states with amplitudes generated from a uniform distribution.

For each state a mixed-dimensional system is used as target architecture. The implementation is written in Python 3, excluding third-party dependencies, and is integrated into the contributions of *Part* IV. The evaluations were performed on a server running GNU/Linux using an AMD Ryzen Threadripper PRO 5955WX (at 4 GHz) and 125 GiB main memory.

The results are presented in Table 9.1, reported in [12] as well. The synthesis procedure is executed 40 times to average possible fluctuations. The initial group of columns provides key information about the benchmark, including the algorithm's name, the number of qudits involved, and the qudit dimensions represented as *Count* $\times$ *Dimension*, randomly selected to further prove the efficacy of the method. For example, an entry 2×4 signifies two qudits, each with a dimension of four. The following two column groups present details on the resulting qudit circuits for both "Exact" (circuits that compile the target state with fidelity 1) and "Approximated 98 %" (compiling the target state with fidelity of at least 0.98). The columns "Nodes" and "Distinct$\mathbb{C}$" represent the average number of nodes and unique complex numbers in the decision diagram representing the state. These are the main metrics for evaluating the efficiency in the decision diagram for storing the mixed-dimensional quantum states. The "Operations" column indicates the average number of multi-controlled operations synthesized for a state on a specific mixed-dimensional architecture relevant for both exact and approximated synthesis. Additionally, the column "Controls" refers to the average median number of controls present in multi-controlled operations synthesized for a state on the mixed-dimensional architecture. The use of controlled operations as a primary metric is justified by the fact that the circuit can later be transposed into a sequence of local and two-qudit operations [177], with also linear complexity in terms of depth, as demonstrated in [178]. Finally, the "Time" column denotes the elapsed time during the approximation and synthesis process. The method is efficient, since the synthesis routine has time complexity linear in the number of nodes of the DD. Moreover, the approximation technique leads to a linear improvement in the number of quantum gates, with the reduction of control nodes during the synthesis.

Both approaches, "Exact" and "Approximated" have been shown to be capable of preparing a given quantum state in a mixed-dimensional system. Due to the regular structure of the first three (non-random) benchmarks, the approximation shows no effect. In contrast, approximation decreases the number of operations (and controls) by about 5 % while losing only 1 % fidelity. Regardless of the approximation step, the individual runs of the benchmarks finish in less than one second.

Overall, the results confirm the successful development of the first automated method for synthesizing quantum circuits that can construct arbitrary states for mixed-dimensional quantum systems. Furthermore, the outcomes demonstrate the method's efficiency and capabilities in optimizing the circuits, preparing the desired state without significantly affecting its fidelity.

Table 9.1 Evaluation of the proposed state preparation approach comparing the average results over 40 runs of the synthesis method per benchmark

Benchmark			Exact (Averaged)					Approximated 98% (Averaged)					
Name	#Qudits	Qudits	Nodes	Distinct$\mathbb{C}$	Operations	#Controls	Time [s]	Nodes	Distinct$\mathbb{C}$	Operations	#Controls	Time [s]	Fidelity
Emb. W-State	3	$[1 \times 3, 1 \times 6, 1 \times 2]$	58.0	5.0	21.0	2.0	0.00	22.0	6.0	21.0	2.0	0.00	1.00
	4	$[1 \times 9, 1 \times 5, 1 \times 6, 1 \times 3]$	1135.0	7.0	49.0	3.0	0.01	50.0	8.0	49.0	3.0	0.01	1.00
	6	$[3 \times 4, 1 \times 7, 1 \times 3, 1 \times 5]$	8657.0	12.0	91.0	3.0	0.03	92.0	12.0	91.0	3.0	0.03	1.00
GHZ State	3	$[1 \times 3, 1 \times 6, 1 \times 2]$	58.0	3.0	19.0	2.0	0.00	20.0	3.0	19.0	2.0	0.00	1.00
	4	$[1 \times 9, 1 \times 5, 1 \times 6, 1 \times 3]$	1135.0	3.0	51.0	2.0	0.01	52.0	3.0	51.0	2.0	0.01	1.00
	6	$[3 \times 4, 1 \times 7, 1 \times 3, 1 \times 5]$	8657.0	3.0	73.0	2.0	0.04	74.0	3.0	73.0	2.0	0.05	1.00
W-State	3	$[1 \times 3, 1 \times 6, 1 \times 2]$	58.0	5.0	37.0	2.0	0.00	38.0	6.0	37.0	2.0	0.00	1.00
	4	$[1 \times 9, 1 \times 5, 1 \times 6, 1 \times 3]$	1135.0	11.0	186.0	2.0	0.03	185.0	10.0	186.0	2.0	0.03	1.00
	6	$[3 \times 4, 1 \times 7, 1 \times 3, 1 \times 5]$	8657.0	14.0	262.0	4.0	0.06	259.0	14.0	262.0	4.0	0.06	1.00
Random State	3	$[1 \times 3, 1 \times 6, 1 \times 2]$	58.0	58.0	57.0	2.0	0.00	40.2	52.9	54.05	1.9	0.00	0.99
	4	$[1 \times 9, 1 \times 5, 1 \times 6, 1 \times 3]$	1135.0	1135.0	1134.0	2.0	0.12	573.92	1071.97	1084.28	2.82	0.10	0.99
	5	$[2 \times 6, 1 \times 5, 2 \times 3]$	2383.0	2383.0	2382.0	4.0	0.14	1255.22	2276.62	2287.07	3.75	0.16	0.99
	6	$[3 \times 5, 1 \times 4, 2 \times 2]$	3266.0	3266.0	3265.0	5.0	0.18	2187.15	3121.65	3136.9	4.79	0.19	0.99
	6	$[3 \times 4, 1 \times 7, 1 \times 3, 1 \times 5]$	8657.0	8657.0	8656.0	5.0	0.44	3176.65	8336.8	8357.35	4.78	0.39	0.99

9.4 Conclusion

In this chapter, we presented a novel synthesis method for state preparation quantum circuits of arbitrary states for mixed-dimensional quantum systems, and the first developed automated method for such an application on quantum architectures with qubits and qudits of different sizes.

The approach uses edge-weighted decision diagrams with a variable number of successors to represent the state and synthesize the quantum circuits. Extended approximation techniques applied to the data structure optimize the number of operations in the circuits. Evaluations demonstrated the method's effectiveness and adaptability in preparing arbitrary quantum states across any mixed-dimensional quantum architecture. Using optimizations substantially decreased the number of necessary operations and controls while maintaining high fidelity. Future work could potentially focus on further refinement and exploration of qudit quantum circuit optimization and approximation strategies, considering the specific capabilities of the targeted quantum hardware.

10 Compilation of Entangling Gates for High-Dimensional Quantum Systems

In order to run the applications described in Section 2.2.4 and running them on a quantum computer, it is essential to have compilation methods capable of translating high-level algorithms into quantum circuits. Mixed-dimensional systems are particularly well-suited for executing some specific classes of quantum algorithms, since a key advantage of qudit systems is their significantly larger set of available instructions compared to qubit systems, allowing for more expressive and efficient implementation of quantum applications. In Section 2.2.2 and Section 8.2.2, we discussed the complexities introduced by high-dimensional systems, particularly regarding quantum operations and entanglement generation. In fact, in two-level systems every entangling gate is equivalent to the CNOT gate [1]. Instead, multi-level systems offer much richer possibilities, including a variety of inequivalent entangling gates. However, these expanded possibilities also introduce challenges, particularly in decomposing quantum operations into the most efficient sequence of local and entangling gates. Additionally, identifying suitable gate sets native to the hardware becomes crucial for compiling entangling gates. Much of the theoretical work needed to address these challenges is still missing. Consequently, no standardized design methods are available, making the compilation of entangling gates a manual process. This limitation restricts the full exploitation of mixed-dimensional quantum systems.

In this chapter, based on [13], we introduce a workflow for compiling arbitrary two-qudit entangling gates in any mixed-dimensional system. The core idea is to map the target unitary operating on two qudits to a single-qudit unitary in an appropriately larger space. This unitary can then be decomposed into two-level operations using established techniques [179]. The resulting decomposition will involve at most d^2 two-level entangling gates of two standardized types. To decompose these standard gates into any hardware-native gate set, we have developed an offline optimization routine, which we demonstrate on a recently developed gate set for trapped-ion qudit quantum processors [9], [43]. In typical experimental scenarios, where native gates act only on a subspace of the full Hilbert space, the cost of this pre-computation is independent of the size of the target unitary. The resulting circuit will express the target unitary using only native gates.

Overall, this methodology enables, for the first time, the fully automatic compilation of entangling gates for mixed-dimensional quantum systems. The proposed method can be applied to any two-qudit unitaries and is efficient, as the numerical optimization step is done offline. Case studies demonstrate the feasibility of the method, and a corresponding implementation is available in open-source as part of the contributions in *Part* IV. The remainder of this chapter is structured as follows: Section 10.1 motivates the problem of entanglement compilation for qudits. Section 10.2 describes the proposed approach to tackle entanglement compilation. Section 10.3 provides case studies on the feasibility of the proposed approach. Finally, Section 10.4 concludes the paper.

10.1 Motivation

Based on the general setting introduced in *Chapter* 2, the compilation of entangling operations into native gate sets is of central importance for efficient quantum computations. The general problem is NP-hard already for qubits [180]; therefore, quantum computers critically depend on efficient compilation methods, if

not optimal. Due to the more complicated entanglement structure of qudits, in comparison with their binary counterpart, the compilation problem is much more challenging upon the consideration of high-dimensional spaces.

10.1.1 Problem Formulation

Qubit circuits may be written using several entangling gates, which are equivalent to the CNOT gate [181]. The situation is very different for qudit systems; as the structure of entanglement becoming much richer, there is not only more freedom in designing quantum circuits, but also more opportunities for optimizing their efficiency through the right set of operations. The challenge for qudit compilation is then the understanding and the harness of this entanglement structure for efficient decompositions. At the same time, the problem has to be computationally tractable.

Example 24. *To make the problem more intuitive for the qubit expert, the use of a metaphor is helpful. Using color coding as an analogy: Information encoded in one qubit can be interpreted as a single grayscale pixel. The information encoded in a* qutrit *($d = 3$) is like an RGB pixel that can combine 3 different colors and all their possible shades inbetween. Now, adding a second pixel, the qubits only have two parameters, where qutrits have 6 parameters derived by leveraging the full potential of the two pixels. This highlights the drastically higher information density in qutrits (and subsequently qudits of even higher dimensions). However, the qutrit, just like the color pixel, requires a vastly more complex control system, in order to efficiently use such a capability.*

The problem of entanglement compilation can then be formulated as follows: *given a unitary U representing an interaction between two qudits of dimension d, find a decomposition of U into arbitrary local unitaries and a pre-defined set of entangling gates, in a way that is as close to the optimum as possible.* In an application setting, the native gate set is defined by the used quantum hardware, which also sets the cost of each component in the decomposition. While the complexity of the optimal solution is generally unknown, the general figure of merit is the gate fidelity given the experimental noise sources for the various gates. Although the discussion will be on qudits of the same dimensionality for the sake of clarity, the method works as well for operations where the qudits are of two different arbitrary dimensions. Moreover, here, we will focus on two-qudit gates, although the methods we present generally apply also to the multi-qudit setting, at the cost of decomposing the tensor product in two-qudit unitaries.

10.1.2 State of the Art

In the current section, we review briefly the existing approaches to compile entangling operations in high-dimensional Hilbert spaces. While some results exist for special cases [179], [182], these are of limited use in actual quantum hardware, which typically does not natively support the types of interactions we might want to work with.

The pioneering work Ref. [179] introduced a synthesis algorithm for qudit entangling gates that produce decompositions through controlled-householder rotations. This enable the proof of a lower and a constructive upper bound on the number of two-qudit controlled-rotation gates required for an arbitrary two-qudit unitary. However, such results apply only to the specific controlled rotations used, without considering the constraints or indeed opportunities of physical qudit quantum processors. In Ref. [183], a compilation algorithm for generic qudit unitaries is presented. The final aim of the mathematical procedure is to reduce the number of gates that use magic state injection protocols. The entanglement complexity of qudit systems is ignored. Although an elegant solution is found, the restricted scalability of the algorithm and the final output, too far from the application to a physical quantum processor, make the work still theoretical. Other previous works [184] try to lay out routines for the synthesizing of qudit circuits, in this particular case for qutrits only.

In this regard, Ref. [182] presents a method to construct circuits for generating different forms of qudit entanglement through the use of a specific gate set, including the controlled exchange gate. While the goal is the generation of specific quantum states from a fixed input, it is still unclear to what extent these methods can be transferred to the more complicated compilation of entangling unitaries.

In Ref. [185], the challenge of multi-qubit compilation is tackled. The authors use parametrized quantum circuits and numerical optimization of the fidelity function, to solve an incremental structure of alternating local and entangling gate layers. More precisely, the work focuses on the compilation of global entangling gates based on an ansatz consisting of alternating global entangling gates and specific equatorial rotations; the approach cannot be applied directly to qudit entanglement compilation, due to the inefficient search strategy. Moreover, the solution provided in [185] is unable to express the wider variety of local and partial entangling operations available in this new setting. Finally, in Ref. [186] arbitrary controlled unitaries are proved to be physically implemented using auxiliary levels in the qudit Hilbert space, regardless of their form.

10.2 Efficient Compilation of Entanglement

Motivated by the challenges described in Section 10.1, we now propose a workflow that enables the compilation of any two-qudit unitary into any target gate set. The considered solution consists of two steps. First, a synthesis method, which targets two-qudit unitary and returns a high-level circuit comprised of only two types of two-level entangling gates, regardless of the initial unitary. This step is crucial to make the technique scalable to arbitrary mixed dimensions. Second, a compilation step that uses parametrized circuits and numerical optimization to decompose the two standardized two-level entangling gates into any target gate set. Although this step can be computationally intensive, it can be pre-computed, as the structure from the first step is fixed. Hence, this step does not affect the scalability of the method.

10.2.1 Step 1: QR Decomposition of Entangling Operations

Qudit quantum hardware typically exploits two-level couplings between the various qudit levels despite having access to a full high-dimensional Hilbert space. This is sufficient for implementing arbitrary single-qudit operations with a cost that is at most quadratic in the size of the Hilbert space, as has been shown in [179]. In the simplest instance, the resulting sequence of operations is composed of two-level *Givens*-rotations between adjacent sites V_i and a diagonal phase matrix Θ in $U = V_k \cdot V_{k-1} \cdot \ldots \cdot V_1 \cdot \Theta$ [179].

This algorithm is universally valid. Yet, it scales quadratically in the Hilbert space dimension, and due to the rigid structure, it can introduce redundant operations. Therefore, for efficiently compiling local qudit unitaries, more advanced algorithms capable of taking into account the structure of the qudits and experimental constraints can be beneficial. For the purpose of the present work, however, it is precisely the standardized structure of the QR decomposition that is beneficial.

In more detail, we first interpret the target two-qudit unitary U as a single qudit unitary in dimension d^2. This is achieved by mapping the two-qudit states to a single qudit as $|i, j\rangle \mapsto |d \cdot i + j\rangle$. Then, we apply the QR decomposition algorithm to that higher-dimensional local unitary. Without loss of generality, we thereby assume a ladder-type coupling, where each of the virtual single-qudit states is coupled to its neighboring states $|j\rangle \leftrightarrow |j \pm 1\rangle$, although the implementation explained in *Part* IV has partially overcome this limitation.

Consequently, the QR decomposition applies successive rotations between the adjacent states in the virtual single qudit.

The result of the decomposition is a set of two-level rotations of the form

$$R(\theta,\phi) = \begin{bmatrix} \cos\frac{\theta}{2} & \sin\frac{\theta}{2}(-i\cos\phi - \sin\phi) \\ \sin\frac{\theta}{2}(-i\cos\phi + \sin\phi) & \cos\frac{\theta}{2} \end{bmatrix},$$

which is followed by a phase matrix to be represented as a sequence of phase rotations on neighboring states, which, in turn, can be decomposed into two-level rotations using the identity $\mathrm{Z}(\theta) = R(\frac{\pi}{2},0)\cdot R(\theta,\frac{\pi}{2})\cdot R(-\frac{\pi}{2},0)$.

As a consequence of our choice of ladder-type coupling, all two-level rotations occur between adjacent states and are thus embedded into the d^2 dimensional Hilbert space as

$$\hat{R}_i(\theta,\phi) = \begin{bmatrix} 1 & \cdots & 0 \\ 0 & \cdots [R(\theta,\phi)]_{i,i+1} \cdots & 0 \\ 0 & \cdots & 1 \end{bmatrix},$$

where the subscript $i, i+1$ denotes the affected states. The resulting matrix is a two-level rotation $\hat{\mathrm{R}}$ embedded in a higher-dimensional space, which translates to an entangling operation in the original two-qudit system.

The next challenge is now decomposing these entangling gates into the native gate set of the target hardware.

A straightforward approach would be to simply try and compile each of the $\mathcal{O}(d^2)$ entangling gates in the sequence into the target gate set, although it is computationally very costly. Instead, one can notice that only two different gates need to be compiled. This is most easily seen as a simple example with two qutrits. By design, each of the two-level rotation R_i is represented as 2×2 block matrices in a $d^2\times d^2$ identity matrix as in Eq. (10.1).

$$\left[\begin{array}{ccc|ccc|ccc} \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & cR & cR & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & cR & cR & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & pS & pS & \cdot & \cdot \\ \hline \cdot & \cdot & \cdot & \cdot & \cdot & pS & pS & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \end{array}\right] \tag{10.1}$$

Here, the horizontal and vertical lines indicate the tensor product structure of the original two-qutrit unitaries. We note that whenever the rotation R_i acts on levels that do not cross these lines, it physically corresponds to a controlled-rotation gate cRot in the two-qudit system (indicated above as cR). When R_i does cross a boundary, it refers to a partial Swap operation pSWAP in the bipartite system (indicated above as pS). While mathematically alike, these operations are fundamentally different in their complexity, so they must be treated separately. Fortunately, it is sufficient to compile just one example of each, since the other cases (for different qudit indices) can be obtained simply by permuting the states of one operation of the same type. For simplicity and generality, we focus on the following gates that act in the 0-1 subspace of the qudit Hilbert space.

After the generation of the abstract sequence of rotations, the compiler simplifies the complexity of the problem by applying permutation gates to every $cRot$ and $pSwap$ gate in order to express the sequence in terms of only two entangling operations, $cRot_{1;0,1}$, $\mathrm{pSwap}_{0,1}$, and permutation gates. Therefore, every operation can be compiled at the expense of compiling efficiently only one $cRot$ and one $pSwap$. The two operations cannot, in general, be implemented directly on the hardware, thus one further step is needed before concluding the synthesis.

Since $cRot_{1;0,1}$, $\mathrm{pSwap}_{0,1}$ are parametrized in θ and ϕ, it would require to compile these default gates for every different value of the two variables. The default gates chosen will be then further decomposed into a sequence of local operations, that will encode the variables θ and ϕ for the equatorial rotations, along with a static entangling gate that will provide the necessary entangling strength and that will act on a two-level subspace.

In such a way, the compiler will follow the decomposition of the controlled rotation $cRot_{1;0,1}$ (control on the level 1 of the first qudit, target the subspace 0-1 of the second qudit) as,

$$\begin{aligned} cRot_{1;0,1}(\theta,\phi) = &[I \otimes R(\pi/2, -\phi - \pi/2)]\cdot \\ &\mathrm{CEX}\cdot[I \otimes Z(\theta/2)]\cdot \\ &\mathrm{CEX}\cdot[I \otimes (Z(-\theta/2)\cdot R(-\pi/2, -\phi - \pi/2)). \end{aligned} \tag{10.2}$$

The decomposition uses the $\mathrm{CEX1};0,1$, which will act on the 0,1 subspace of the second qudit, while the partial swap:

$$\begin{aligned} pSwap_{0,1}(\theta,\phi) = &[H \otimes H]\cdot \mathrm{CEX}\cdot[H \otimes H] \\ &[I \otimes R(\pi/2, -\phi - \pi/2)]\cdot \mathrm{CEX}\cdot \\ &[I \otimes Z(\theta/2)]\cdot \mathrm{CEX}\cdot[I \otimes Z(-\theta/2)]\cdot \\ &[I \otimes R(-\pi/2, -\phi - \pi/2)]\cdot \\ &[H \otimes H]\cdot \mathrm{CEX}\cdot[H \otimes H] \end{aligned} \tag{10.3}$$

and the $\mathrm{CEX}_{1;0,1}$ is used once again in the pSwap operation.

Example 25. *Let us consider the case where two qutrits interact in order to better understand the intermediary compilation step, the operations involved, and the new encoding. The controlled rotation on levels $|7\rangle$ and $|8\rangle$ appears in the decomposition, which correspond to $|21\rangle$ and $|22\rangle$ in the two-qutrit system. Since the method requires the controlled rotation to be applied to the 0-1 subspace, the permutation of the levels of the two-qutrit system is required. We add local rotations with angle π and phase $\pi/2$ that will act as permutations and route $|21\rangle$ to $|10\rangle$ and $|22\rangle$ to $|11\rangle$.*

The decomposed sequence will be:

$$\begin{aligned} cRot_{2;1,2}(\theta,\phi) = &(P_{1,2} \otimes P_{0,1}\cdot P_{1,2})^{\dagger}\cdot \\ &cRot_{1;0,1}(\theta,\phi)\cdot \\ &(P_{1,2} \otimes P_{0,1}\cdot P_{1,2}), \end{aligned}$$

where $P_{i,j}$ denotes a permutation of levels i and j.

Note that after executing the controlled-rotation, the sequence concludes in the uncomputation of the permutation sequence, to restore the original encoding for the subsequent operations.

At this stage, the synthesis returns a sequence of local operations that will encode the angles of rotation and the permutation of the states, interleaved with the chosen CEX gate. Such is the default choice made in the project, yet the user could provide a different one that could act on a different subspace. Then, the compiler will now proceed with the last step: the compilation of the CEX gate with an entangling gate input by the user and compatible with the target machine.

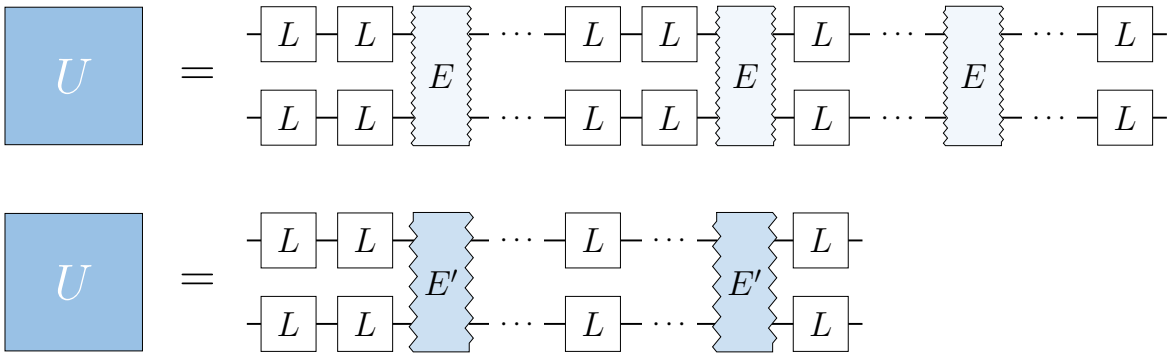


Figure 10.1 Given a two-qudit unitary U, compilation results depend on the structure and the noise inherent to the chosen gate E; new local (L) gates will match the gate E.

10.2.2 Step 2: Layered Compilation

This section describes the second step of the compiler and the lowest abstraction level before getting to a physical implementation. The software component takes as input the target unitary to compile and the target entangling gate for the decomposition. The entangling gates could be either a primitive entangling gate of the target quantum hardware or a higher-level abstract gate. The software outputs a compiled sequence composed of arbitrary local two-level rotations and the chosen gate.

At the core of the second component are parametrized quantum circuits that, once solved for different gates, lead to different decompositions. More precisely, as Figure 10.1 shows, results present different depths and noise characteristics, related to the noise and power of the single entangling operation.

The section will continue with a brief overview of parametrized quantum circuits, followed by a detailed explanation of the construction of the problem representation and its resolution by the layered compiler.

Parametrized Quantum Circuits (PQCs)

PQCs consist of a series of quantum gates dependent on a set of continuous or discrete parameters θ_i that can be optimized. The particular initial choice of the sequence of gates is called an *ansatz*. Such a circuit is then optimized, typically by a classical algorithm, in order to minimize a cost function that encodes the solution of interest.

Such an approach can be used for a range of computational questions [187], like Hamiltonian ground-state energy estimation [188] or compilation [185]. In the following, we discuss the important design choices during the ansatz construction, its resolution, and the ansatz evolution during the optimization loop.

Construction of Ansatz and Expressibility

The first step consists of choosing the building blocks for our ansatz and then incrementally composing them in order to create the final circuit to be solved. There are two types of operations related to in the quantum circuit: local operations and entangling operations.

The initial objective is to find a representation for single-qudit operations that is parametric and that achieves maximal expressibility [189] with a minimal number of parameters. In this case, expressibility refers to the circuit's ability to generate a large fraction of two-qudit unitaries. The proposed solution exploits theoretical results [190] already developed in the literature. The single qudit lives in a d-dimensional $(d \geq 2)$

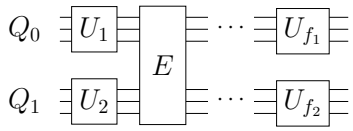


Figure 10.2 Example of an ansatz. Each layer contains an entangling gate, which is preceded by a generic unitary U on each one of the qudits and followed by two more generic unitaries; in this manner, every layer is universal.

complex Hilbert space $\mathcal{H} = \mathbb{C}^d$ spanned by the orthonormal basis $|0\rangle, \ldots, |d-1\rangle$. Local qudit unitaries will be written as in [190], in which elements of the special unitary group $\mathcal{SU}(d)$ are parametrized as:

$$\begin{aligned} U = &[\prod_{m=0}^{d-2} (\prod_{n=m+1}^{d-1} \exp(iZ_{m,n}\lambda_{n,m}) \exp(iY_{m,n}\lambda_{m,n}))]\cdot \\ &[\prod_{l=1}^{d-1} \exp(iZ_{l,d}\lambda_{l,l})] \end{aligned}$$

using $d^2 - 1$ real parameters $\{\lambda_{m,n}\}$ in the interval between 0 and π for $m > n$, $\pi/2$ for $m < n$, as well as 2π for $m = n$.

On this space, Z is a diagonal trace-less operator, a generalized form of the Pauli σ_z applied to the subspace spanned by $|m\rangle$ and $|n\rangle$, while Y is the generalized form of the Pauli σ_y applied to the subspace on the subspace spanned by $|m\rangle$ and $|n\rangle$.

$$\begin{aligned} Z_{m,n} &= |m\rangle\langle m| - |n\rangle\langle n| && \text{for } 0 \leq m < n \leq d-1 \\ Y_{m,n} &= -i|m\rangle\langle n| + i|n\rangle\langle m| && \text{for } 0 \leq m < n \leq d-1 \end{aligned}$$

Such a formulation is chosen to be expressible for the single qudits and it has the minimum number of parameters required for representing the special unitary group.

The second objective is the efficient use of user-specified entangling gates in the ansatz compilation. The usage of multiple species of qudit entangling gates inside the same ansatz will be discussed in future research. Although the user can specify the preferred choice for the entangling primitive, here, we focus on using a single entangling gate, choosing between two gates commonly used in the trapped-ion platform: the Molmer-Sorenson (MS) gate in Eq. (10.4) [9] and the generalized light-shift (LS) in Eq. (10.5), a genuine qudit entangling gate [43], [191].

$$MS(\theta) = e^{-i\frac{\theta}{4}\cdot(I\otimes I + \sigma_{x_{01}}\otimes\sigma_{x_{01}})} \tag{10.4}$$

$$LS(\theta) = e^{-i\theta\cdot\sum_{i=0}^{d-1}|ii\rangle\langle ii|} \tag{10.5}$$

These are both well-established operations in quantum computing hardware with well-characterized noise characteristics. Furthermore, they operate in a distinct manner (the light-shift acts on phases, the MS acts on populations) with different entangling power [43] when applied to qudit systems.

Example 26. *The compiler solves an ansatz like the one discussed in Figure 10.2.— Consider a two-qudit entangling unitary U. Figure 10.1 illustrates two different compilation results: (1) The top result requires multiple pre-selected entangling gates with low entangling power but potentially less noise. (2) The bottom result uses two previously selected entangling gates with high entangling power but more noise per gate.— Such a selection of the best result heavily depends on the performance of the gates, i.e., specifics of the underlying hardware. Both cases will always produce a correct result since more entanglement can be built up with additional gates; entanglement generation can also be reduced through the judicious use of single qudit gates between entangling layers.*

Solution of the Ansatz

The optimization of the ansatz works similarly to a variational quantum algorithm. In each step, the full circuit is simulated through the product of the gate matrices for the chosen parameters. This is computationally feasible for two-qudit gates in dimensions of practical relevance, given current and future quantum hardware capabilities. For multi-qudit gates, the applicability of this approach is expected to be limited. The resulting unitary matrix in each optimization step is then compared with the target matrix in accordance with an objective function. Here, the choice function was the fidelity between the unitaries [192], for two-d-dimensional systems:

$$\textit{Fidelity}(A, B) = \frac{1}{d^2} \operatorname{Tr} \langle A^\dagger \, , B \, \rangle$$

For the purpose of this compilation task, fidelity has several desirable properties for physical applications compared to other frequently used objective functions for matrix optimization.

The optimization loop is performed through the iterative simulation of the ansatz and verification of the fidelity achieved by the current solution of the optimization algorithm.

Optimizer

The optimization step can be performed by means of heuristic methods, as seen in Section 2.3.4. The employed optimization method is the dual annealing method, meant as a combination of classical and fast simulated annealing, coupled with the L-BFGS [193] algorithm on boxed constraints, a local search method applied in the neighborhood of the solutions found by the global method in the high dimensional landscape of the problem to optimize. We refer the reader to Ref. [194] for details on this algorithm. Alternatively, due to the unclear trainability of qudit PQCs, the use of gradient-based methods was outside the scope of the work.

Binary Search

The final resulting circuit should have the least number of layers that can compile with the desired fidelity of the entangling gate, although the cost of a circuit depends on its depth and gates structures. The maximum number of layers for the search is heuristically chosen as $2 \cdot d^2$. Such a number was found to be sufficient for generating maximally entangled qudit states from the least powerful operation, acting only on two fixed physical levels. The compiler uses a binary search, and the number of layers is decreased iteratively as the gate can be decomposed for a certain number of layers under a heuristically chosen time period. Otherwise, if the target fidelity is not reached or the optimizer does not converge before the end of a timer, the number of layers is increased.

10.3 Case Studies

The considerations made so far in the chapter have already shown the complexity of entanglement compilation in mixed-dimensional systems. The solution proposed above is meant to provide a step forward in the field of compilation for qudit systems. The implementation is open-source and integrated into the contributions presented in *Part* IV. It is completely written in Python 3.8, with the exception of the external dependencies Numpy [195] and Scipy [194]. This section provides corresponding case studies, in order to indicate the feasibility of the workflow and its usefulness in compiling any multi-level entangling operation. The use cases focus on the compilation of the controlled-sum gate CSUM, a characteristic operation in qudit systems discussed in the previous sections.

Table 10.1 Results of the workflow on CSUM for dimension 9 and dimension 16

System	Dim.	cRot	pSwap	CEX_{tot}	MS_{tot}	LS_{tot}	$(1-F)$
two qutrits	9	44	24	184	1472	368	$< 10^{-4}$
two ququarts	16	92	36	328	9184	10168	$10^{-3} \sim 10^{-4}$

Although it is characterized by two compilation steps, the computational complexity of the workflow is dictated on a high level by three routines. The first routine is the QR decomposition applied on the unitary to compile; the algorithm has quadratic complexity in the number of dimensions of the two-qudit interaction with size D, therefore $\mathcal{O}(D^2)$. The result is a sequence of at most D^2 operations. The second routine is the translation of every output operation in terms of local operations and CEX; this algorithm has complexity $\mathcal{O}(D^2)$ because it is linear in the number of QR output operations. Finally, the last routine of the workflow is the solution by optimization of the ansatz, which has complexity of order $\mathcal{O}(\log D)$. In fact, the last compilation step is a binary search, with each step of duration linear in the number of dimensions of the original unitary.

The evaluations were performed on a server running GNU/Linux using an AMD Ryzen 9 3950X and 128 GiB main memory. The layered compilation step is performed under a time limit of hours heuristically chosen as $d/4$ (e.g., 4 h for a two-ququarts unitary), at each step in the binary search. The compiler has minimum target infidelity ($1 -$ *Fidelity*) of 10^{-3}. Several runs of the optimization algorithm could lead to better results, and beyond dimension 16, the computational time and the fidelity are affected.

The considered example follows the default strategies encoded in the compiler. Each controlled rotation and partial swap is transformed by local rotations into $cRot_{1;0,1}$ and for the partial swaps $pSwap_{0,1}$. The automatic choice for generating entanglement is the controlled exchange gates CEX. More precisely, $\mathrm{CEX}_{1;0,1}$ has control on the first level of the first qudit and the targets on the first two levels of the second qudit.

The designer has the possibility to either decide to follow a different path and impose a decomposition for which they have a specific implementation in terms of the involved entangling gate, or to compile it directly for the machine.

In the studies, we considered the implementation of the CSUM gate for qutrits and ququarts. Table 10.1 shows the results. Between the near-term usable physical qudit dimensions [9], the solution for ququarts proves an already difficult and useful design case. The columns denote the dimension ("Dim."), which is followed by the number of controlled rotations, and partial swaps, as well as the total number of CEX, MS, and LS gates required for the decomposition, respectively. The last column gives a bound on the achieved infidelity (i.e., $1 -$ *Fidelity*). In the first phase, the synthesis of CSUM for 9 dimensions (two qutrits) requires 20 controlled rotations and 6 partial swaps; the synthesis on two quarts (dimension 16) outputs 42 controlled rotations and 6 partial swaps. These decompositions are not necessarily optimal, by construction. The controlled rotations are further decomposed into 2 CEXs and the partial swap into 4 CEXs, with the resulting sequence being general for all dimensions.

In the smaller case, we compile the entangling gate in terms of two native gates used in the latest ion trap qudit processors [9] [43] and introduced before, the MS and LS gate, in order to show to the adaptability of the workflow. The manual decomposition of the CEX gate dedicated for the two-qutrit case by itself is non-trivial, even for an experienced quantum information specialist. The inherent difficulty arises not only from the amount of intuition needed but also from the inability to easily generalize the single result found to higher dimensions. The pen-and-paper optimized sequence for CEX in dimension 9 is composed of two

LS gates with angle π, while two MS gates are also required to perform the same CEX at any dimension, if the sequence is allowed to exploit auxiliary levels [9].

The layered compiler performs a similar decomposition with the same number of π-LS gates autonomously. Without auxiliary levels, the compiler outputs a sequence of 8 MS gates. For both the compilation results, the infidelity reached is below 10^{-4}. In the case of two-ququarts, optimized sequences could not be achieved, whereas the layered compiler autonomously decomposes the CEX with infidelity between 10^{-3} and 10^{-4}, at the cost of 28 MS gates and in an alternative of 31 LS gates. The number of gates required for a two-ququarts circuit indicates the increased complexity of the problem compared to two-qutrits.

The results are in line with the expected *automation overhead* in a first automatic solution. In fact, it was predictable that the used ansatz would ask for an over-parametrization [196], or using more layers than necessarily, which allows for a great simplification of the landscapes by eliminating spurious local minima. Although the results are not expected to be efficient to the point of practical applicability on a quantum system, this is only one component of a complete compiler for mixed-dimensional systems; as such, future developments will comprise optimization routines[14], [197]–[200].

In summary, the results show the feasibility of the proposed workflow to deliver decompositions efficiently for any two-qudit unitary. Moreover, the results provide evidence of the complexity in the compilation of entangling operations for multi-level systems, especially compared to binary systems. This chapter investigated one of the first key steps towards the full automated compilation for qudits.

10.4 Conclusion

The challenges for efficient and reliable application of entangling gates inside qudit circuits arise because of the complexity in understanding the amount of entanglement generated by an operation as well as the corresponding affected levels—commonly referred to as structure. Consequently, the operations are mostly considered as black-boxes with the question of whether it is possible to reach an implementation for a certain qudit platform and with a given gate-set. So far, the study of feasibility and the implementation of entangling operations for specific qudit technologies was performed by quantum information specialists manually, without the promise of re-utilizing a particular decomposition for a system certain number of dimension, to those of greater dimensionality.

In this chapter, we introduced a complete workflow for compiling any two-qudit unitary into any target gate set. While the resulting gate sequences are typically not optimal in terms of gate count, the method is computationally efficient, since the only step involving numerical optimization can be pre-computed. The case studies confirm the feasibility of the workflow as well, for the first time, the automated results of compilation of generic entangling operations to any dimensionality. The implementation is available as open-source, and the steps for using the compilation procedure will be presented in *Part* IV. In the future, the proposed approach may be improved up on by utilizing auxiliary qudit levels, alternative ansatz designs, compression of synthesis results, and circuit optimization in post-processing.

11 Adaptive Compilation of Multi-Level Quantum Operations

Mixed-dimensional systems, with their larger state space, are particularly well suited for representing integer-valued states, making them useful for quantum simulations of higher-dimensional theories. To develop a complete compiler for mixed-dimensional quantum systems, it is essential to efficiently compile single-qudit operations. These compilation methods serve as the final component, enabling the automatic translation of high-level quantum algorithms into quantum circuits. This eliminates the need for error-prone manual conversion while complementing the compilation of entangling operations discussed in *Chapter* 10.

Automated methods and data-structures must be developed to decompose unitary operations into sequences of two-level operations that respect the physical constraints of the multi-level–qudit platforms, as explained in Section 2.1. When such decomposition is not entirely feasible, the methods should focus on finding good-enough solutions as efficiently as possible, using heuristics as outlined in Section 2.3.4.

In this chapter, based on [14], we develop an efficient adaptive compiler for single-qudit operations. To this end, we first introduce a corresponding representation of underlying physical options and constraints in terms of an *energy coupling graph*. Based on this, we propose adaptive methods that aim to leverage this representation to efficiently compile arbitrary unitaries. In particular, this method exploits the free placement of logical information on the underlying physical information carriers to achieve a significant improvement in the complexity of the compiled operations. To showcase the benefits of the proposed method, we compiled large sets of unitaries of dimensions 3, 5, and 7 to different target architectures—confirming the flexibility to adapt to arbitrary coupling structures of different multi-level systems as well as experimental constraints. Although the cost function used is inspired by the state-of-the-art qudit hardware platform [9], the methods presented are general and apply to any physical platform and cost function. Finally, a comparison to the state-of-the-art static QR decomposition showed a significant improvement in the resulting costs on average, with further potential to trade off worst-case costs and run-time. The resulting tool has been made available as open-source and integrated into the contributions of *Part* IV.

The remainder of the chapter is structured as follows: Section 11.1 contains the motivation for this chapter. In Section 11.2, we introduce the proposed algorithm alongside the required cost function. Section 11.3 evaluates the proposed approach. Finally, Section 11.4 concludes the paper.

11.1 Compiling Multi-Level Quantum Operations

The definition of an arbitrary (multi-level) quantum operations follows the basics of quantum information processing, as reviewed in the background sections. However, the elementary operations available in state-of-the-art quantum computing hardware typically couple only two levels at the time. Furthermore, they may be subject to additional physical or practical constraints. This natively implies the availability of only a small set of single-qudit operations, while the vast majority require compilation into elementary operations. The goal of a good compiler is then to not only realize the desired computations, but also to achieve so in the most efficient way by fully exploiting the potential of the available technology. Solutions designed for compiling unitaries on qubit systems are generally directed towards the scope of scalability issues and

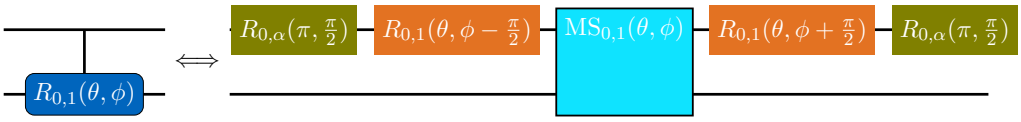


Figure 11.1 Example of local operations on entangling operation.

error mitigation. In a first step, the focus of compilation lies on single-qudit operations. The reasons are three-fold:

1. A main advantage of qudit systems is that complicated qubit entangling operations are traded for simpler qudit local operations. As such, a powerful local compiler is critical for exploiting this potential.

2. Qudit systems provide the possibility of performing computations between quantum systems of different dimensions and different characteristics [22]. By construction, this can not be achieved through qubit systems where all the units are computationally identical to each other.

3. The compilation of entangling operations heavily relies on the use of local unitaries. Therefore, a local unitary compiler, such as the one we present, is necessary for compiling general multi-qudit unitaries.

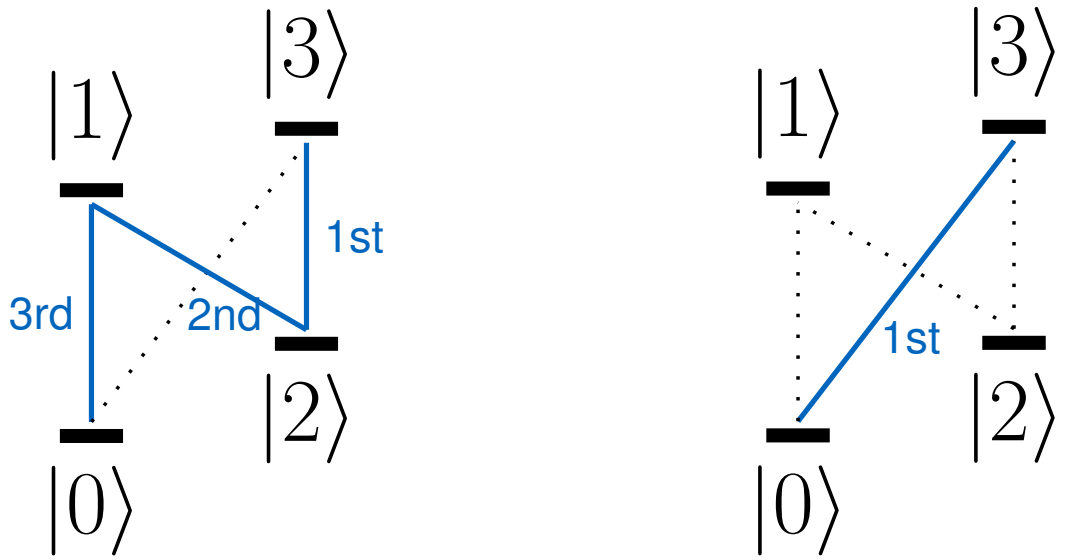
Example 27. *Figure 11.1 displays an example of an entangling operation performing a controlled arbitrary rotation; there is a total of four unitaries and one entangling gate, in this case the Mølmer–Sørensen (MS) gate [201]. In order to ensure the correct behavior of the MS gate as a CNOT gate, additional local gates are required on each one of the qudits involved in the operation, which are applied just after the native operation. First, $d-1$ operations per qudit are needed to correct intrinsic phase shifts [9]. Second, another batch of 4 local gates turn the corrected MS gate into the desired CNOT gate. Thus resulting in a total of 10 local operations for compiling one entangling gate on two ququarts (4 levels). It is evident that the ratio of local gates per entangling gates is rather high. Furthermore, not being able to compile efficiently each one of the local unitaries is an obstacle to scalability in this type of systems once full applications are deployed.*

The state-of-the-art qudit compilation [9], [42], [179], [202]–[205] relies on a straight-forward *QR decomposition algorithm*, which is employed to decompose a matrix into an orthogonal matrix and an upper triangular matrix. In such an algorithm, the sequence of elementary two-level operations is fixed a priori; only the angles of the rotations are calculated as they depend on the target unitary. It is important to note that this method does not take care of particular restrictions encountered in real-world systems. An example is the impossibility of directly performing certain rotations and shorter paths for routing two levels close to each other. Therefore, an arbitrary operation will, in most cases, have a better decomposition than the one generated by the static sequence. Hence, the blind application of more operations could lead to greater error-rates. As illustrated by the following example, the state of the art does not overall fully exploit the potential of current multi-level architectures.

Example 28. *Consider a target unitary U that rotates between $|0\rangle$ and $|3\rangle$. The QR decomposition will produce three valid operations to realize the rotation. Namely, U will be decomposed into $R_{2,3}, R_{1,2}, R_{0,1}$. Figure 11.2a illustrates the static sequence.*

Given the connectivity in Figure 11.2, a shorter (and optimal) decomposition $R_{0,3}$ exists and makes immediate use of the existing connection, but is not found by the QR approach. Figure 11.2b illustrates the optimal decomposition. The static sequence requires three reordering pulses. Conversely, the optimal sequence will be composed of only one gate—thus highlighting the possible improvements over the state of the art even in small examples.

The objective is to develop a method that is adaptive and generalizable to arbitrary coupling structures. Due to physical constraints, every technology has however restrictions that have to be taken into account



(a) QR decomposition

(b) Optimal solution

Figure 11.2 Comparison between QR decomposition (3 steps) and optimal solution (1 step) to rotate between $|0\rangle$ and $|3\rangle$

during compilation. In order to make these restrictions transparent, we propose: a dedicated representation of options; constraints in terms of an *energy coupling graph*; an adaptive algorithm that enables us to control a number of trade-offs due to the technology that implements the qudit logic, as well as the influence of the sources of error. Compiling in this work can be viewed as a multi-tiered procedure that requires several techniques to apply a single arbitrary operation as efficiently and reliably as possible. Moreover, an adaptive algorithm often leads to decompositions. Hence, it reduces the cost of application for a given unitary or a sequence of them.

11.2 Adaptive Compilation of Multi-Level Quantum Operations

This section provides a detailed description of the proposed approach. To this end, we first introduce the concept of an *energy coupling graph* that describes the possible physical connections in the architecture as well as the corresponding subset used for the actual compilation. Hence, we describe an adaptive decomposition approach which explicitly considers the potential of underlying technology when compiling a given unitary.

11.2.1 Energy Coupling Graph

The way information is encoded in the qudits (such as, which energy levels represent which basis state) has a significant impact on the quality of the resulting compilation result. Various technologies have been employed to implement certain quantum operations for encoding information [9], [22], [23], [25]–[31], [33], [34], that leverage different physical platforms with different degrees of freedom. By construction, they inherently use multi-level physical structures to represent their state, albeit only two of those levels are currently commonly used. Constructing a representation of these levels is compulsory for understanding how to efficiently access them and relate to each other for performing operations.

The connections between physical levels are not arbitrary. They are, in fact, heavily dependent on the technology employed. In order to capture the possible connections, we use the term *coupling* for levels of

a qudit, in analogy to the coupling describing possible interactions between qubits in two-level quantum computers. More precisely, a coupling between levels of a qudit indicates the possibility of transitioning from a certain level to a second one, caused by an impulse to the system. Moreover, we model the possible couplings between levels with a graph—which we refer to as *energy coupling graph*. Consequently, the restrictions imposed by the energy coupling graph are referred to as *energy coupling constraints*. The nodes of the graph represent different energy states within the physical information carrier, and this is what inspires the term. The transitions can be mathematically indicated with unitary matrix operators and associated with corresponding gates. In terms of the two-level Bloch sphere, an operation is a rotation on the sphere with the considered energy levels at the poles. By that, the resulting energy coupling graph represents a possible mapping of states to levels as well as the corresponding resulting constraints as provided by recent physical realizations [205].

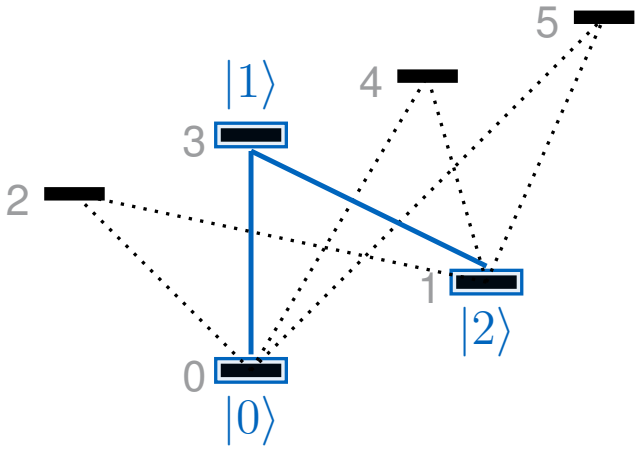
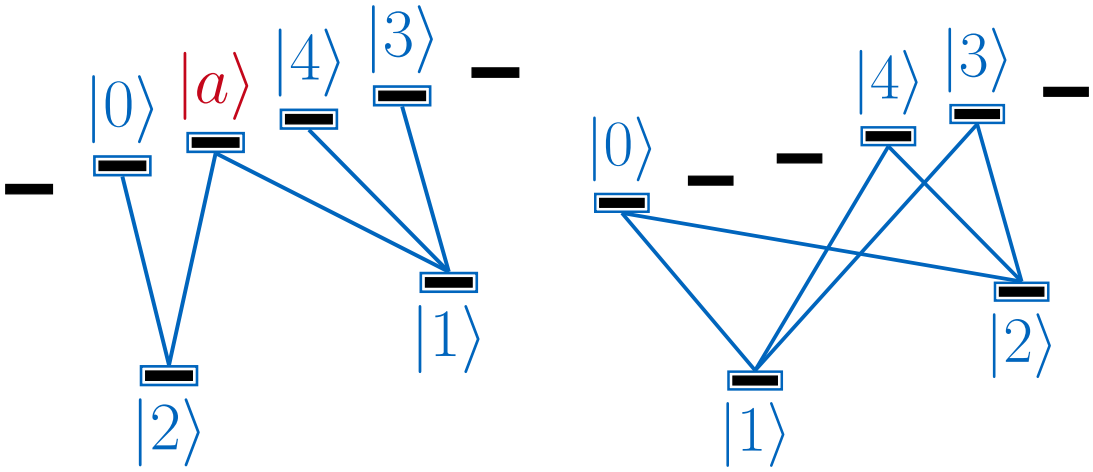


Figure 11.3 A qutrit energy coupling graph

Example 29. *Figure 11.3 visualizes the proposed energy coupling graph for a single three-level qudit, or qutrit. The black horizontal bars refer to the physical levels provided by the technology, and the black dotted lines show the theoretically possible couplings. The blue rectangles indicate the logical nodes of the graph mapped to a subset of the physical levels, and the blue solid lines are the edges of the constituted graph that are built on the theoretically available coupling. The blue lines represent the physically feasible transitions or an efficient subset of them.*

With the energy coupling graph introduced, we can graphically inspect the three main differences between the theoretically possible connectivity in a qudit and the realistic connections in a quantum system. Given the graph of each possible level available from the technology, we start by assigning the logical states to the physical levels—referred to as *mapping*. Future work considers exploiting the energy level graph with the scope of automatically providing an initial mapping efficiently, while the current solution takes the mapping as input from the designer. Typically, not all the physical levels in the energy coupling graph are used to encode logical states, as some of them may be used for ancillary tasks, such as routing two distant logic states or as a cache memory, temporarily storing information. By principle, in the ideal qudit, there are transitions in an all-to-all fashion between all the logic states. However, in the non-ideal real world, only certain physical transitions can be effectively used—thus enabling only certain rotations in the qudit state-space. Additionally, it is possible that logically contiguous states are placed on distant nodes of the graph. For instance, it may happen when optimizing some particular applications on the qudit architecture. In this chapter, we lay the foundations for a new layer of abstraction that acts as an interface between a physical qudit and a corresponding compiler.



(a) Realistic Qudit **(b)** Ideal Qudit

Figure 11.4 A realistic qudit, limited by physical complexity, compared to an ideal qudit.

Example 30. *Consider Figure 11.4a, which shows realistic qudit defined by three main differences compared to the ideal case depicted in Figure 11.4b.*

- *Figure 11.4a has fewer connections than the ideal case, which may be implied by physical (e.g., atomic selection rules), or practical constraints.*
- *In Figure 11.4a, the logic states are not placed in order. For example, the logic state* $|2\rangle$ *is connected to* $|0\rangle$*, but not to* $|1\rangle$ *and* $|3\rangle$*.*
- *Each of the two graphs is connected, yet Figure 11.4a has an ancillary state* $|a\rangle$ *to ensure connectivity.*

The energy coupling graph, as proposed, offers designers automatic tools and means to properly reflect the potential as well as the restrictions of the underlying technology. In the following, the energy coupling graph is used to guide the compilation.

11.2.2 Decomposition

As the energy coupling graph is assumed as described in the previous section, the next step in the multi-tiered compilation process is a *decomposition* of the target unitary U. In the scope of this chapter, the decomposition algorithm is intended to produce as output a sequence V_i of two-level rotations between adjacent physical levels in the energy coupling graph and a diagonal matrix Θ of arbitrary phases [205] as in Eq. (11.1):

$$U = V_k \cdot V_{k-1} \cdot \ldots \cdot V_1 \cdot \Theta \tag{11.1}$$

As such, the objective of the algorithm is to output a sequence of operations that realize the target unitary U, while taking into account the capabilities and restrictions of the underlying technology.

Example 31. *Consider performing a rotation between two not directly connected levels. Some naive solutions for improving the flexibility of the QR decomposition to different graphs could be either to permute the unitary term, in order to match the encoding on the graph, with a reordered basis set,* or *to fix the encoding and add single-qudit reordering pulses to the sequence to bring logic states in the energy coupling graph closer, perform the operation and then bringing them back. Both variants work in a* static *fashion: potentially introducing unnecessary operations and, consequently, unnecessary costs.*

Algorithm 6 Adaptive Decomposition

Require: Tree.root: n, U, Energy coupling Graph:g, Cost Limit: cl
Ensure: Decomposition, Best Cost

```
if checkDiagonal(n.U) then return
father = n
for c do
    for r1 ≥ c do
        for r2 > r1 do
            if U_{r1,c} ≠ 0 and U_{r2,c} > threshold then
                θ = 2 · arctan[|U_{r2,c}/U_{r1,c}|]
                φ = −(π/2 + arg[U_{r1,c}] − arg[U_{r2,c}])
                c', πgates, g' = Cost(U, g, R_{r1,r2}(θ, φ))
                U' = R_{r1,r2}(θ, φ) · U
                if father.cost + c' < cl then
                    father.addNode(gates, U', g', c', cl)
for child in father.children do
    Adaptive(child)
```

We propose an *adaptive algorithm* based on a recursive and bounded depth-first search and using a tree structure to keep track of the exploration of the possible decomposition, with the aim of alleviating the drawbacks of static decomposition approaches. An overview is provided in Algorithm 6. By adaptively changing the energy coupling graph, this algorithm avoids the problems characterizing the static solutions in Example 31. The algorithm starts with the allocation of only the root of the tree, where the only information stored are the cost limits imposed by the static QR decomposition, the target unitary to compile, and the initial energy coupling graph. The cost limits for the adaptive approach are the costs of the QR decomposition applied on the same target unitary, plus the summed costs of each single rotation of the adaptive decomposition. The complexity for calculating this cost bound is asymptotically quadratic in the number of dimensions. In every step, the algorithm applies a two-level Givens-rotation [9], [202] to the unitary and checks if the resulting matrix is diagonal. In that case, the algorithm returns the nodes in the tree that are on the path of the best decomposition and terminates. Otherwise, the code enters three nested loops: one over all the columns, and two over indices that keep track of two different rows, where the innermost loop's index is always greater than the first loop over the rows. At every step of the execution of these loops, the entries of the matrix indexed by the column counter and the two rows indices are selected, and they are compared against two constants (indicated as 0 and *threshold*). The angle of rotation is computed as a ratio between the two entries and the phase is calculated similarly with a ratio of the two components of each of these complex numbers. Therefore, the thresholds guarantee that the coefficients will not diverge to infinity due to the usage of floating point numbers and possible division by a number close to zero. Furthermore, the threshold should be chosen in relation to the experimental performance of the fundamental two-level couplings, such that the resulting parameter fluctuations are negligible compared to the inherent imperfections of the quantum operations. Once the integrity of the entries is verified, the angle and phase of rotation are obtained as shown in the pseudo-code. These two parameters are then used to create the Givens-rotations of the decomposition as elementary blocks. These matrices indicate the rotation on the Bloch sphere of two energy levels, but they are also the rotation of a subspace of the larger Hilbert space of the qudit.

The rotation between two levels i and j are expressed as

$$R_{i,j}(\theta,\phi) = \exp\left(\frac{-i\theta}{2}\Big(\cos(\phi)\sigma_x^{i,j} + \sin(\phi)\sigma_y^{i,j}\Big)\right) \tag{11.2}$$

where $\sigma_{\{x,y\}}$ are the Pauli-$\{X, Y\}$ matrices, describing the two-level coupling, θ is the rotation angle, and ϕ is the phase of the rotation.

The cost of each rotation is calculated through a method that considers the necessary routing of the states on the graph of the qudit into account, as detailed in Section 11.2.5. Moreover, the reordering sequence that enables the rotation, and the modified graph are returned alongside the calculated cost. If the total current costs of the decomposition are smaller than the cost limit, then the rotations, the cost, the new unitary rotated, and the new graph are saved as new child of the current node in the tree. The function recursively calls itself with each child of the current node as a parameter. As the implemented method is in the category of backtracking algorithms, the worst-case complexity is of asymptotically exponential order [206]. However, the experimental results show that a suitable choice of the cost function allows the algorithm to perform similarly to the static QR decomposition.

11.2.3 Satisfying Energy Coupling Constraints

The decomposition from the previous section leads to a sequence of operations that are elementary on the respective technology but not necessarily in line with the level-to-state mapping, i.e., transitions between states might not be immediately possible. The problem of satisfying the energy coupling constraints is addressed by inserting reordering gates into the operation sequence and by introducing rules for correct patterned sequences.

Routing

The decomposition is driven by the cost of the possible logical rotations on the transition of two logical states. In this case, the energy coupling graph is used to track the position of each logical state. Considering a Givens-rotation on the logic states i and j, the returned routing sequence is a list of reordering gates that will bring the logical state j adjacent to the logical state i in the energy coupling graph. The reordering gates are constructed as Givens-rotations with default values $\theta = \pi$, $\phi = -\pi/2$, and indices of rotation associated with the nodes on the shortest path from i to j. The energy coupling graph is adapted accordingly. Algorithm 6 considers the energy coupling constraints.

Graph rules

Another challenge arises from the fact that the reordering pulses are not permutation matrices, as they carry additional sign flips. This is a result of the mathematical form of the generators of the primitive physical rotations. In the following, we outline a set of simple rules meant to be applied to the reordered gate sequence. First, rotation matrices are defined to rotate from a lower level to a higher level. In case, during compilation, we encounter a situation where this ordering is reversed, this leads to an inversion of the sign of the phase ϕ for that operation. Second, the sign of θ is flipped whenever a gate in the sequence is preceded by a reordering pulse whose higher level coincides with any of the levels of the gate in question. The last rule leverages a particular feature of the graph, where each node is able to store a phase value that the physical level has accumulated during computation. The feature is useful for solving two challenges: the first one arises when compiling several unitaries in a row where the Z-rotations that are not expressed in the form of a gate can be just recorded in the graph's nodes, thus allowing for even shorter sequence and avoiding the problem of propagating these gates explicitly; the second challenge is unique to the adaptive

algorithm where routing sequences are not uncomputed as in the QR decomposition case. The sign flips from the reordering pulses do not cancel and, therefore, need to be tracked explicitly as an accumulation of π phases due to the routing gates. Consequently, the third rule adds the phase stored on the higher energy node to the phase value of a gate; then it subtracts the phase stored on the lower energy node.

Ancilla states

The proposed approach also enables the possibility of the inclusion of ancillary states in the graph. The use of ancillary states is known to enable improvements in gate complexity [22] and to simplify the application of established quantum gates in qudit Hilbert spaces [9]. Once a state is declared as ancillary, the algorithm tracks them like regular states, but also gains the option of exploiting them for routing purposes or caching. Furthermore, in Example 32 the input shows how ancilla states can be used for compiling two independent unitaries in parallel on the same qudit.

Example 32. *Consider the matrix in Eq. (11.3), which is composed of two different portions. The upper-left part is the Hadamard gate which we want to compile on the first 3 logic levels. The lower-right part is the matrix that will operate on two ancillary states. It is, therefore, possible to compile two separate operations in one unitary.*

$$\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 1 & e^{\frac{2\pi i}{3}} & e^{\frac{-2\pi i}{3}} & 0 & 0 \\ 1 & e^{\frac{-2\pi i}{3}} & e^{\frac{2\pi i}{3}} & 0 & 0 \\ 0 & 0 & 0 & a_{11} & a_{12} \\ 0 & 0 & 0 & a_{21} & a_{22} \end{bmatrix} \tag{11.3}$$

11.2.4 Phase Shift Propagation

After the decomposition and the insertion of reordering gates from the previous sections, the phase shifts have to be taken into account. They are are encoded in the diagonal matrix Θ (see Eq. (11.1)). In order to ensure the correctness of the overall decomposition, they have to be decomposed into individual phase shifts on single levels, albeit the phase shift are not physically realized. These individual phase-shifts are formulated as diagonal matrices, with only the i-th entry on the diagonal set to a phase shift of ϕ of the logic state, and 1 otherwise, with $i \in \{0, \ldots, d-1\}$.

Example 33. *Consider a three-level qudit system and a rotation of ϕ on the second entry on the diagonal. The corresponding matrix is shown in Eq. (11.4).*

$$R_{Z,1}(\phi) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & e^{i\phi} & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{11.4}$$

At the end of the circuit a phase shift is negligible, since changes in phase immediately before a measurement do not change the outcome probabilities. Then, these phase shifts can be introduced at zero cost in the sequence, as they can be propagated to that region. We refer to these rotations as “virtual

Z-gates"[1] since they are not executed on the quantum computer. An example of the commutation of phase gates and Givens-rotations is shown in Eq. (11.5).

$$M \cdot R_{0,1}(\theta, \alpha) = R_{0,1}(\theta, \alpha - \phi + \gamma) \cdot M \tag{11.5}$$
$$\text{with } M = \begin{bmatrix} e^{i\phi} & 0 & 0 \\ 0 & e^{i\gamma} & 0 \\ 0 & 0 & e^{i\delta} \end{bmatrix}$$

The equation is given for a rotation on the coupling $R_{0,1}$. Yet, it can generalized to any coupling. The application of the commutation can lead to a linear cost reduction but is potentially limited once entangling gates are introduced into the circuit.

11.2.5 Cost function

The previous sections described the steps involved in the decomposition of the target unitary, satisfaction of the energy coupling constraints, and propagation of the phase shifts. The decisions made in these steps are justified by a cost function, which is provided in this section. This cost function is inspired by the trapped-ion system recently presented in [9] and the numerical results derived from benchmarking the platform in the same reference. Yet, it is primarily determined by considerations that are found in a similar form also in other technologies. More precisely:

- The cost of each rotation scales linearly with the rotation angle.
- Each two-level coupling is calibrated for a fixed rotation angle, typically $\pi/2$. Different angles will incur additional cost due to the non-ideal performance of physical devices. We assume this cost to increase linearly with the distance from the calibrated value.
- The system is initialized almost perfectly in a fixed initial state. The couplings that do not involve the initial state will incur a higher cost (we assume here a factor that is a function of the distance of two logic states in the graph) since they require a more complicated multi-step calibration procedure.

Taking into account these effects, we model the cost of the single rotation as

$$C_1 = 10^{-4} \cdot d(i,j) \cdot \left(4\theta + \left| \mod\left(\theta + \frac{1}{4}, \frac{1}{2}\right) - \frac{1}{4} \right|\right) \tag{11.6}$$

for $\theta \geq 1/4$ (in units of π). The overall factor 10^{-4} in the equation comes from the fundamental cost of a π-rotation [9].

Besides the bare cost of logical rotations, there is an additional cost associated with reordering the logical states on the coupling graph. This is implied by the fact that it is, in general, not fully connected. This is done with a sequence of π-rotations, with a cost again governed by Eq. (11.6). Then, these two terms constitute the full *experimental cost* of a decomposition.

11.3 Evaluation

The proposed method of adaptive compilation utilizes the energy coupling graph with the aim of finding more efficient decompositions. We developed an implementation to evaluate the method and compared it with the state-of-the-art static-sequence algorithm based on the QR factorization [9]. The implementation

[1] In this case, the term virtual Z-gate is not associated to the Pauli operation.

Table 11.1 Evaluation of costs for the QR and the proposed (adaptive) decomposition

		QR decomp. [9]			Proposed decomp.		
Dim	Unitaries	min	avg	max	min	avg	max
3	333	8	25.47	140	4	21.69	36
	333	4	21.19	28	4	9.49	12
	333	12	35.89	44	4	14.93	20
5	2 985	60	166.34	200	64	113.36	124.63
	2 985	36	101.27	128	16	82.86	91.05
	2 985	88	240.62	280	16	121.97	133.7
7	6 438	132	662.35	756	44	255.58	279.08
	6 438	192	518.23	620	48	389.94	425.51
	6 438	88	392.35	508	44	330.3	361.51

Cost is multiplied by 10^4 to aid legibility.

will be presented in *Part* IV. We compile sets of random Clifford-unitaries [162], suitable for randomized benchmarking to account for different scenarios, since these gates average all errors to depolarizing noise. The solution proposed is general enough to compile any gate set. Still, the choice to compile Clifford gates is justified in several other ways. In fact, being able to compile the Clifford group proves a seamless integration capability of the solution with error-correcting schemes. Moreover, Clifford gates belong to the family of unitary t-designs [47], with t equal to 2, able to approximate the performance of non-Clifford operations. In particular, their decomposition complexity is well approximated. The gate set is compiled for dimensions 3, 5 and 7 to three target architectures each. The choice of dimensions is coherent with the useful dimensions inside the latest qudit processors [9]. However, new systems will require the compilation of larger unitaries that the compiler is able to decompose efficiently with a small effort in elaborating more sophisticated cost functions. The architectures are different in the number of nodes, connectivity, and mapping of the logical states on the physical level, thus resulting in nine different target architectures. Diagonal unitaries are excluded from the sets, since they are decomposable in a sequence of virtual phase-gates and the cost of the decomposition in each algorithm is zero. The evaluation was performed on a computer with an Intel i7-10750H and 32 GB of memory, running Ubuntu 20.04 and Python 3.8.

The results are presented in Table 11.1. The first column gives the dimensionality, followed by the number of considered unitaries in the second column. The next three columns show the costs of the QR decomposition for the considered unitaries, listing the minimal, average, and maximal costs according to the cost function in Section 11.2.5. The same listing on costs are contained in the next three columns for the adaptive compilation.

Comparing the costs in Table 11.1 shows a clear advantage of the proposed adaptive algorithm. In every case, the maximal cost of the considered set of unitaries is lower compared to that of the QR decomposition—in some cases there is an improvement of up to a factor of 3. The same conclusion can be drawn from

the average costs as well, which is considerably lower in each set of considered unitaries. Regarding the run-time, each decomposition of a unitary in this evaluation took less than 1 s for the QR decomposition and the adaptive approach, respectively. Hence, by spending the same computational efforts, we get significantly better compilation results.

Moreover, the proposed method can be configured to get even better results. In the evaluations summarized above, we set the cost limit (see Algorithm 6) to 1.1 of the cost of the QR decomposition to have an acceptable run-time for the thousands of unitaries. For a more realistic scenario, where a designer wants to realize an individual unitary, decreasing the cost limit will push the proposed algorithm to get even smaller costs—at the expense of longer run-time. Furthermore, the improvement attainable by the proposed adaptive method of course depends on the unitary that is considered, but also on particularities of the underlying technology. One important factor is how dense the connectivity of the energy levels is, where a denser (and more physically unrealistic) connectivity leads to less improvement on the cost by the adaptive algorithm. With the implementation of the proposed method being publicly available, every designer can accordingly use these features to get better results than the ones summarized in the extensive evaluations covering thousands of unitaries from above.

Overall, the results confirm that, using the proposed method, more efficient decompositions can be determined and, additionally, further option exists to trade-off worst-case costs and run-time—providing a solution that aims at fully exploiting the potential provided by the given technology.

11.4 Conclusion

In this chapter, we proposed an *adaptive compilation* algorithm to attain efficient decompositions of arbitrary local unitaries for multi-level quantum systems. Compared to binary quantum systems, qudits feature a much richer state-space, allowing for different placements of logical information under consideration of practical constraints, such as the energy coupling constraints. With the constraints and options represented in an *energy coupling graph*, the proposed algorithm exploits the available potential by flexibly adapting the mapping between logical states and physical energy levels to enable a much more efficient compilation. Indeed, turning the argument around, the proposed algorithm provides a way to compare different placements of logical states by means of the average cost to decompose random Clifford unitaries. This flexibility of the proposed algorithm to adapt to realistic constraints in the possible couplings as well as the cost of different rotations enable significant improvements in gate costs.

12 Summary of Part III

This part of the thesis addressed the critical role of compilation in quantum computing, particularly as quantum hardware scales in size and complexity, and applications demand increasing precision and efficiency in resource consumption. The focus is on mixed-dimensional quantum architectures, which integrate qubits and higher-dimensional qudits, posing unique challenges for compilation. Efficient compilation ensures that high-level quantum programs can be translated into low-level instructions optimized for specific quantum devices, minimizing errors and maximizing hardware capabilities. The immense search space makes this an extremely challenging problem to solve optimally, particularly when attempted manually. The solution not only consumes significant time, but also introduces a high risk of errors. The results in this part of the thesis collect successes from across the compilation stack and show the benefit of using data structures like decision diagrams for compilation and the advantages of using heuristic methods as search methods for compiling quantum circuits. The results are the first example of compilation for mixed-dimensional quantum computers and form a baseline for future work. In this part, the following contributions were presented:

- *Chapter* 8 presented a method for compressing qubit circuits into qudit circuits. This allows algorithms originally designed for qubits to be restructured and mapped on hardware that utilizes higher-dimensional qudits. This adaptation reduces the number of non-local operations–minimizing unwanted noise in quantum computations. These positive results showed the effectiveness of the method and indicated potential secondary use of the tool as a resource estimator technique to evaluate mixed-dimensional architectures and enhance the execution of qubit circuits.

- *Chapter* 9 introduced an innovative synthesis approach for state preparation in quantum circuits that target arbitrary states within mixed-dimensional quantum systems. The method utilizes edge-weighted decision diagrams with a variable number of successors for state representation and quantum circuit synthesis. By employing advanced approximation strategies on the data structure, the gate count is significantly optimized in the circuits. This method proves highly efficient and versatile for the preparation of arbitrary quantum states in any mixed-dimensional quantum setup, representing the first automated approach applicable to quantum architectures comprising qubits and qudits of varying dimensions.

- *Chapter* 10 introduced a universal workflow for compiling any two-qudit unitaries into any hardware gate set that supports two-level entanglement gates. Current qudit entangling gates require manual, system-specific implementation, the method offers an algorithmically efficient approach, with pre-computable numerical optimization. Although the gate-count is not optimal, case studies demonstrate the workflow's practical feasibility across different quantum systems, showing high flexibility and overcoming the difficulties of compiling two-qudit entangling gates on a mixed-dimensional architectures.

- *Chapter* 11 presented an algorithm aimed at efficiently breaking down local unitaries in multi-level quantum systems. Since common decomposition algorithms return sequences with lots of redundancies, the new adaptable method enhances gate decomposition costs while also taking into account realistic coupling constraints and single rotation expenses.

The contributions in this part enable practical quantum computations on mixed-dimensional systems, allowing researchers and developers to focus on high-level problem modeling without being constrained by hardware-specific complexities. The tools and methodologies developed in this part have been integrated into an open-source software framework described in *Part* IV, providing the quantum computing community with accessible resources to further advance the field and collaborate.

Part IV

Open Source Software and Languages

13 Resulting Open-Source Framework and Language

In this Ph.D. thesis, we have discussed how efforts have intensified with the development of universal qudit quantum processors achieving competitive performance [9], [30], [37]. In Section 2.1, we explained how these technological platforms operate, and in Section 2.2, we discussed how to exploit these platforms for performing quantum computations. The computational model that emerges is what we refer to as mixed-dimensional quantum computing, which is already being employed in quantum applications where such an approach is more naturally suited [66], [73], [74]. Section 2.2.4 provides a detailed list of applications that clearly benefit from these architectures, suggesting that this approach could pave the way for enhanced quantum computing, where qudits and qubits work synergistically.

Qubit-based quantum computing has significantly benefited from advanced and well-established software tools, which are widely implemented on larger systems. In contrast, while progress in quantum hardware and the demonstration of applications for mixed-dimensional quantum computing have been promising, a critical challenge remains: the lack of dedicated quantum software for mixed-dimensional systems. This scaristy assures that the vast potential of qudit-based systems remains largely untapped.

To fully realize the capabilities of mixed-dimensional quantum computing, urgent development of software solutions is needed. Such tools must address the design challenges of circuits, enabling algorithms to run efficiently, reliably, and at scale. Without these advancements, the potential for scientific breakthroughs is restricted as workforce efforts are diverted toward implementation challenges instead of chasing novel applications and discoveries.

In this chapter (based on [15]), we introduce MQT Qudits, an open source framework for mixed-dimensional quantum computing, which is part of the *Munich Quantum Toolkit* (MQT, [4]). MQT Qudits provides efficient, automated, and accessible methods for tackling challenging problems in the design of quantum computing applications. In the following, we will provide a brief overview of MQT Qudits from both a user's perspective (showcasing how to utilize the tool for describing problems and algorithms in the form of quantum circuits, and how to simulate and compile them) and a technical perspective (covering the underlying methods, the choices made during development, and how to contribute to the framework). More precisely, we will:

- first, walk through the advantages, challenges, and opportunities of mixed-dimensional quantum computing and discuss how the framework can address these aspects (Section 13.1),
- illustrate how the design of the framework and how to get started with MQT Qudits (Section 13.2),
- introduce the language and interfaces used by the tool (Section 13.3), and
- present the corresponding methods for quantum circuit simulation (Section 13.4) and quantum circuit compilation (Section 13.5).

13.1 Why a Framework for Mixed-Dimensional Quantum Computing?

To provide the necessary background for this discussion, we refer to Section 2.2, where the foundational concepts of mixed-dimensional quantum computing are introduced. We now turn our attention to the advantages, challenges, and opportunities presented by mixed-dimensional quantum computing.

Current qubit-based quantum computers offer potential advantages in solving highly complex problems by reducing the amount of computational resources, yet scalability remains challenging. Qudit-based quantum computers could mitigate these limitations to some extent by increasing the information density and computational efficiency for a given number of quantum information carriers.

For instance, qudits enable native encoding of mixed-dimensional problems, such as those often found in quantum simulation tasks. Rather than requiring $\lceil \log(d) \rceil$ qubits to encode a d-dimensional object, thereby turning simple two-body into complicated many-body interactions, the mixed-dimensional approach enables us to allocate only the required resources and maintain simple circuit structures. In some cases, this may even allow for the flexible truncation of certain quantities to seamlessly enable the computation at different levels of accuracy [74]. However, these varying degrees of freedom become part of the problem and have to be tracked. Being able to describe the problems easily and efficiently is the first step towards the exploitation of emerging mixed-dimensional quantum processors.

The introduction of qudits in quantum computing architectures creates a lot more freedom for design choices and problem-tailored solutions. This includes not only the choice and evolution of the single-qudit encoding but also the wide range of possible local and entangling operations. While for qubits the controlled-NOT (CNOT) gate is the central resource, qudit systems can build on an ever-increasing collection of non-equivalent entangling gates with varying degrees of entangling power [207] to enable efficient circuit design. The flip side of this greatly increased freedom and flexibility is the increased complexity of designing short and efficient, noise-resilient quantum circuits for qudit systems. Finding a way through this landscape of advantages, challenges, and opportunities presented by mixed-dimensional quantum systems necessitates a comprehensive set of methodologies and automated software tools for effective exploration and utilization, such as classical simulators and compilers.

To this end, the MQT Qudits framework offers an integrated advanced language for quantum circuit representations, simulation, and compilation methods, and software that facilitates the utilization of mixed-dimensional quantum systems. This framework enables researchers to systematically exploit the potential for quantum information processing tasks across diverse technology platforms and to access the full potential of mixed-dimensional qudit systems.

Given the benefits and necessity for a unified software framework for mixed-dimensional systems, we can now explore the structure of MQT Qudits and see how it relates to the previous chapters in this thesis. Afterward, we will show how to use the library and how each component was engineered.

13.2 Framework Design and Components

As illustrated in Figure 13.1, the MQT Qudits framework implements a comprehensive software architecture for mixed-dimensional quantum computers. The framework's workflow, depicted through blue interconnected blocks, demonstrates the system's modular pipeline structure. On the left side of the diagram, we observe the domains of the supported problems, including high-dimensional quantum simulation and integer optimization. The user interaction layer, shown in the adjacent block, provides two primary interfaces: DITQASM, which will be explored in Section 13.3.2, and a Python interface. The central portion of the diagram illustrates the core data structures and methods, which are among the biggest accomplishments of this thesis, and have been implemented to enable the functioning of the framework. We could call it the

"core" of the software framework, encompassing smart graphs, machine learning components, heuristics, tensor networks, and decision diagrams.

As shown in the right section of the figure, the processing pipeline diverges into two primary paths: a simulation track (upper block) and a compilation track (lower block). The simulation component of the framework, implemented through a series of smaller components, includes a backend system, noise modeling, and specialized processing simulation engines. These leverage both the C++ ad hoc decision diagram simulator and external libraries for mixed-dimension tensor networks, both covered in Section 2.3, and respectively *Part* II. On the other hand, the compilation track, whose methods have been covered in *Part* III, is a compiler composed of passes that can solve several tasks with varying levels of optimization, including state preparation and different types of compilation of qudit operations.

The overview concludes with the showcase of supported platforms, shown in the rightmost section, which extend to both trapped ion and superconducting quantum systems, with specific integration for the trapped ion machine of the Qudits group at the University of Innsbruck. The QR codes visible in the diagram link to specific framework components that were polished as standalone tools in the past for validation of the research results: MQT-misim, MQT-qudit-compression, MQT-qudit-entanglement, and MQT-qudit-single. These provide access to individual modules of the framework, most of which are seamlessly integrated into MQT Qudits.

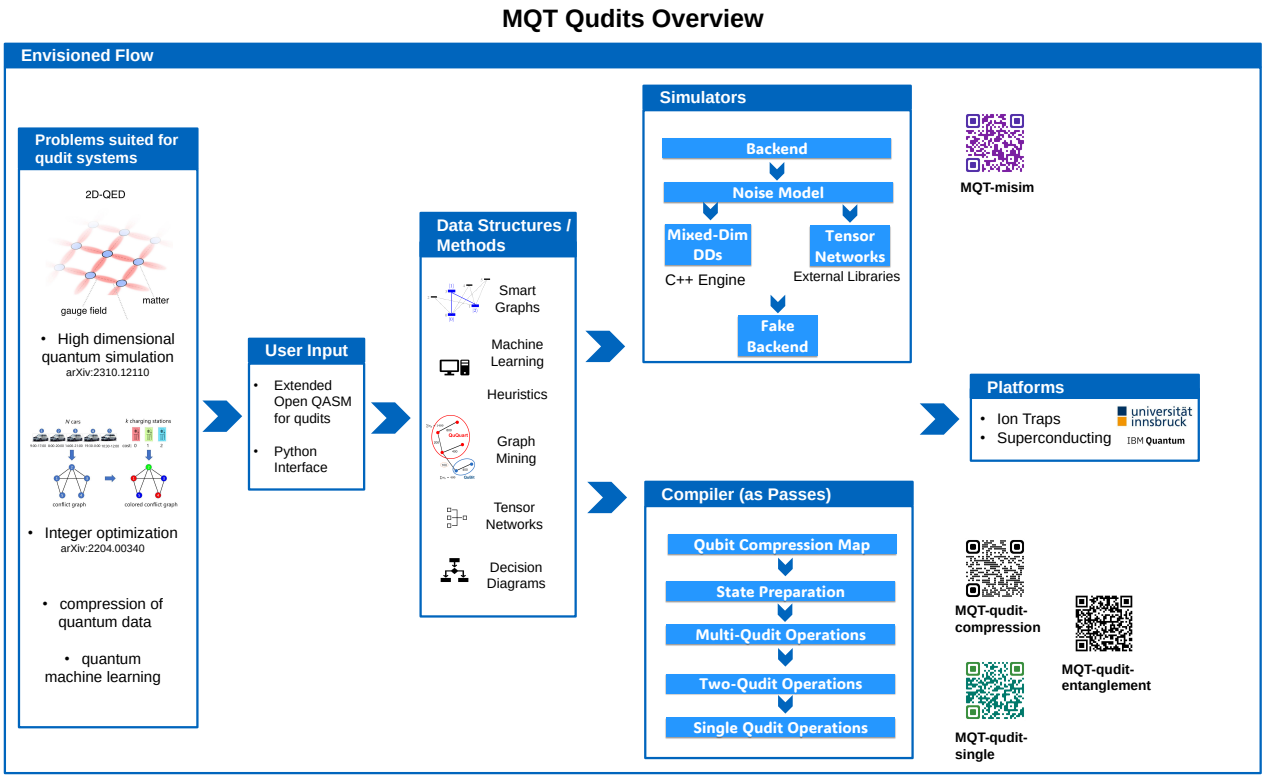


Figure 13.1 A visual representation of MQT Qudits in one single chart.

After reviewing the design of the system and its components, we can now describe how to install the framework and take the first steps in using MQT Qudits.

13.2.1 Getting Started

MQT Qudits empowers users to efficiently create, simulate, compile, and enhance mixed-dimensional quantum circuits across diverse quantum computing platforms. In the next sections, we provide a concise

overview of how to set up and begin using MQT Qudits. The library is primarily written in Python, with components such as simulation engines written in C++ for performance reasons and then interfaced with Python. The installation process is designed to be user-friendly and does not require explicit compilation of the source. MQT Qudits supports Windows, Linux, and Mac for all versions of Python from 3.9 to 3.12. Installing MQT Qudits simply requires the execution of the following lines:

```
$ python3 -m venv .venv
$ source .venv/bin/activate
(.venv) $ pip install mqt.qudits
```

The code above demonstrates how to create a new environment and subsequently install the library directly from the terminal. After setting up the working environment, it is possible to write your first program.

The remainder of the chapter will introduce each component of the framework from two different perspectives: one focusing on the functionality for users, and the other on the technical aspects for motivated scientists and developers. We will always begin with the former. The next section will illustrate how to kicks start the framework and develop.

13.3 Languages and Interfaces

The first step towards simulating and compiling mixed-dimensional quantum circuits, which could potentially solve complex problems, is to define a language that can represent them. None of the existing languages provide appropriate means for this purpose.

13.3.1 Programming Languages for Quantum Computing

A pivotal development that enabled the exploitation of classical technologies was the development of programming languages that would allow users to abstract and disregard many of the mechanisms at the system level, focusing on the algorithmic development and not how aspects like register allocation or transistor-level arithmetic operations are implemented. A great success story coming from the realm of classical systems. Consequently, as quantum computers continue to evolve, the development of programming languages tailored to quantum systems has become crucial. Although the possibilities of programming quantum systems on a higher level of abstraction are still limited, these languages still enable researchers and developers to express quantum algorithms and circuits effectively. This section provides an overview of quantum programming languages, with a particular focus on quantum assembly languages, like the widespread OpenQASM.

Similarly to classical systems, quantum programming languages can be categorized into several levels of abstraction:

1. High-level languages (e.g., Q# [208], Silq [209])
2. Mid-level languages (e.g., Quipper [210], Scaffold [211])
3. Low-level languages (e.g., OpenQASM [212], Quil [213])
4. Hardware-specific assembly languages (e.g., Pyseq [20])

It is relevant to illustrate the core features of each abstraction layer to also comprehend what are the different needs of a user that are being satisfied. High-level languages focus on the programmer productivity, they have automatic qubit management and garbage collection, built-in quantum primitives and operations.

This type of languages leverage better the classical ecosystem and tools, such as the possibility of strong type systems, compile-time checks and support for classical-quantum hybrid programming with optimization algorithms and conditional subroutines. Mid-level languages present a balance between abstraction and hardware control. In fact, there is a focus on circuit manipulation and optimization capabilities, which is much more specific than quantum algorithms subroutines. Circuits can be generalized thanks to the support for parameterized quantum circuits and template-based circuit generations.

Nowadays, the features of high-level and mid-level languages are blended within software frameworks such as Qiskit [3] or Pennylane [8], where the objective is to support the knowledgeable user in constructing highly efficient quantum circuits for larger-scale quantum computers, with the minimum development time. These software frameworks are often accompanied by quantum resource estimation tools [214], [215], which connect the design of high-level circuits and their implementation by estimating the cost of subblock and recommend the user in which subblock to use or which ones should be better implemented.

In the lower end of the languages stack, low-level languages enable a direct hardware agnostic quantum circuit description, with explicit gate-level programming, limited abstraction facilities, and basic classical control flows based on mid-circuit measurements. Lastly, hardware-specific languages are tailored to specific quantum hardware with direct control of physical quantum operations. Practically, this translates into quantum control software with timing and pulse-level control. This level of expression enables hardware-specific optimizations that can leverage the biased noise of specific architectures or exploit the multi-level structure of the quantum system. Logically, this ad-hoc tailored software has very limited portability.

The rest of the section will primarily focus on low-level languages, particularly quantum assembly, which has been the abstraction used for researching, compiling, simulating, and executing quantum circuits during the majority of the pre-fault tolerant quantum computing era.

Quantum Assembly Languages

Quantum assembly is a whole family of languages for programming quantum computers at the low-level, in an imperative way, with manual management of memory, manual choice of the most suitable operations and results storage.

Examples of QASM languages that demonstrated crucial in the development of the first quantum platforms are:

- Quil [213]: Developed by Rigetti, used with their PyQuil framework.
- Blackbird: QASM-like language used for continuous-variable (CV) quantum computing in Strawberry Fields [216].
- cQASM [217]: used primarily by the quantum compiler QuTech and its platform.

Although the family is wide and varied, OpenQASM (Open Quantum Assembly Language [212]) is one of the most widely used and established. Developed by IBM, it provides a flexible and extensible framework for expressing quantum circuits. Although the language has evolved in time and provides new complex feature in the latest releases, here we reference and treat the analysis of OpenQASM 2.0. The basic syntax of OpenQASM allows the user to apply some actions that can be hardware-agnostic for the vast majority, like declaring:

- **Quantum Registers**: Allows declaration of quantum registers.
- **Classical Registers**: Supports classical bits for measurement results.
- **Gates**: Defines a set of universal quantum gates.

- **Measurements**: Provides operations for measuring qubits.
- **Control Flow**: Supports conditional statements and loops.
- **Subroutines**: Allows definition and use of quantum subroutines.

Example 34. *Here's a simple example of an OpenQASM program that creates a Bell state, one of the fundamental entangled quantum states. The program initializes two qubits in a separable state, applies quantum gates to create entanglement, and then measures the results. When measured, both qubits will always yield the same result (either both 0 or both 1), demonstrating quantum entanglement.*

```
OPENQASM 2.0;
include "qelib1.inc";

qreg q[2];
creg c[2];

h q[0];
cx q[0],q[1];
measure q -> c;
```

This program:

- *Declares a quantum register* `q` *with 2 qubits*
- *Declares a classical register* `c` *with 2 bits*
- *Applies a Hadamard gate to the first qubit*
- *Applies a CNOT gate with the first qubit as control and the second as target*
- *Measures both qubits and stores the results in the classical register*

Advantages, Challenges and Limitations of Quantum Assembly After observing the different languages and the different needs they tackle, it is relevant to analyze some of the main advantages. For instance, their hardware agnosticism allows users to work independently of specific quantum devices, guaranteeing portability and broad applicability across platforms, differently from hardware-specific languages. Moreover, they provide fine-grained control over quantum operations, enabling developers to tailor circuits precisely to optimal performance. They present interoperability that facilitates a smooth exchange and translation of quantum circuits between various tools. This has been proven to be a pillar in promoting collaborations, comparisons, and integration between quantum software and tools across the quantum ecosystem.

However, these benefits come with non-negligible challenges. Quantum assembly languages are verbose, requiring significant effort to encode algorithms manually, which can be time-consuming and error-prone. On top of that, it is complex to describe quantum algorithms concisely, requiring users to have a deep understanding of quantum gates and circuits. On top of that, the agnosticism is only apparent since they still demand hardware-specific knowledge, which poses a steep learning curve. As quantum systems scale up, this representation might become a bottleneck for compilers, fault-tolerant protocols, and automated decision-making.

After reviewing the landscape of languages for quantum computers and the different families of languages, with their benefits and disadvantages, we introduce the new programming language of MQT Qudits:

DITQASM, which belongs to the family of QASM languages. In the next section, we explain the syntax of it and how it is interpreted in the framework.

13.3.2 User's Perspective

MQT Qudits introduces DITQASM, an adaptation of OpenQASM [212] for describing mixed-dimensional quantum circuits and architectures. QASM languages serve as a bridge between high-level quantum algorithms and their physical implementation on hardware, and in this case, DITQASM introduces new fundamental features, as illustrated in Figure 13.2. Users can use DITQASM to translate abstract quantum algorithms into executable forms, analyze circuit complexity, and study the impact of noise and error correction in real quantum systems. Key features include:

- Register allocation: descriptions of the quantum and classical registers involved and their dimensionality.
- Gate-level description: the precise specification of quantum gates and their application to qudits of different dimensions, with a new way of controlling gates on arbitrary levels.
- Hardware-agnostic representation: a standardized way to express quantum circuits across different quantum computing platforms.
- Customizability and extendibility.

Example 35. *Figure 13.2 shows the DITQASM implementation of a quantum program using two qudit registers: one containing two qudits with dimensions 2 and 3, and another containing two qudits with dimensions 4 and 7 (lines 2-3). Additionally, a classical register is declared with 4 units (line 4). The circuit is then constructed, beginning with the inverse of the Hadamard gate ("h") applied on the first qudit of register 2 and controlled by the state of the two qudits in the other register. This control is described using the "ctl" syntax (line 5), a new feature introduced in DITQASM. This is followed by a controlled-sum gate "csum" [13] (line 6) and two subspace rotations "rxy" [9]. The definitions of the gates do not require any knowledge of the technology platform implementing the multi-level quantum systems, and the dimensionality is not required at the application of the gates, but the focus is only on the semantic information of the rotations. The measurement is performed according to the default options (lines 9-12).*

Alternatively, circuits can be described by a Python interface, as shown in Figure 13.3. Here, the Python program starts with the import of the two main components of a quantum circuit: quantum registers, a quantum circuit, and, optionally, classical registers (lines 1-3). After creating a quantum circuit instance, the next two lines allocate two qudit registers, and a classic register (lines 5-11): "`reg_1`" and another one named "`reg_2`." The ability to name and allocate different registers facilitates the direct translation of interactions, such as those represented on a lattice or a graph, into an algorithm. Subsequently, the quantum operations are applied: a Hadamard gate on the "`reg_2`" register and a controlled-sum gate, which generalizes the CNOT gate (lines 12-14). The interpreter tracks the dimensionality of the qudits and applies the corresponding operations. The program concludes with a measurement of the qudits.

The Python interface and DITQASM also support various gates, including hardware-specific ones like the Molmer-Sorenson (MS) gate from [9] and the generalized light-shift (LS), a genuine qudit entangling gate [43], [191], as well as custom unitary evolutions on one, two, or multiple qudits and qubits.

13.3.3 The Technical Perspective

The section gives an intuition of the functioning of the DITQASM interpreter and pointers for further extensions and improvements. The distinguishing feature of DITQASM is its capacity to define arbitrary

```
DITQASM 2.0;
qreg reg_1 [2][2, 3];
qreg reg_2 [2][4, 7];
creg meas[4];
inv @ h reg_2[0] ctl reg_1[0] reg_1[1] [0,0];
csum reg_2[0], reg_1[0];
rxy (0, 2, pi, pi/2) reg_1[1];
rxy (0, 1, pi, pi/2) reg_2[1];
measure reg_1[0] -> meas[0];
measure reg_1[1] -> meas[1];
measure reg_2[0] -> meas[2];
measure reg_2[1] -> meas[3];
```

Figure 13.2 Example of DITQASM representing a program run on a quantum architecture made of two registers, with qudits of dimensions 2, 3, 4, and 7.

```
from mqt.qudits.quantum_circuit import QuantumCircuit
from mqt.qudits.quantum_circuit import QuantumRegister,\
                                       ClassicRegister

circuit = QuantumCircuit()
reg_1 = QuantumRegister("reg_1", 2, [2, 3])
reg_2 = QuantumRegister("reg_2", 2, [4, 7])
meas = ClassicRegister("meas", 4)
circuit.append(reg_1)
circuit.append(reg_2)
circuit.append_classic(meas)
circuit.h(reg_2[0]).control([reg_1[0],\
                                 reg_1[1]], [0, 0])
circuit.csum([reg_2[0], reg_1[0]])
circuit.r(reg_1[1], [0, 2, np.pi, np.pi/2])
circuit.r(reg_2[1], [0, 1, np.pi, np.pi/2])
```

Figure 13.3 A simple quantum program written for a mixed-dimensional quantum computer, in Python, through MQT Qudits.

Hilbert space dimensions for individual elements within a quantum register, a capability previously unavailable. The expanded gate set, equipped with automatic dimensionality matching, makes the language hardware-agnostic. Furthermore, the automatic generalization of single-qudit and entangling gates to higher dimensions abstracts away the internal structure and properties of individual qudits, facilitating a focus on algorithm development. The language supports fine-grained control over multi-level systems, allowing for the specification of arbitrary control levels within the range $[0, d_i - 1]$ for the control qudit in controlled operations. Classical registers are implemented as integer wrappers, maintaining consistency with the multi-level quantum paradigm.

The current Python 3-based DITQASM interpreter is in its nascent stages. The current version (0.3.0) employs pattern matching during the parsing of the text of the quantum program; the instructions will be collected, and all the information and metadata will be tracked in the software. This data is then used to generate all the corresponding components and eventually compose them in the correct fashion. This approach ensures extensibility, allowing for easy incorporation of additional gates and features as the field evolves.

The Python interface and DITQASM are in direct relation to each other. This is sufficient for a local prototyping environment but will not suffice for large-scale deployments or for using MQT Qudits in a quantum computer-centric HPC environment. Possible solutions would involve the development of a suitable *Quantum Intermediate Representation* (QIR) for mixed-dimensional systems. Prospective developments include the integration of classical control flow syntax and mid-circuit measurements, crucial for implementing error correction schemes in qudit systems. These advancements are particularly pertinent given the growing relevance of qudit-based approaches across various domains of quantum computing research.

13.4 Quantum Circuit Simulation

As seen before in *Part* II, quantum circuit simulators are essential in assessing the quality of quantum circuits and quickly iterating new design choices for the algorithms to run. They enable the exploration of alternative qudit-based approaches for various applications, the verification of quantum circuits, the validation of noise models, the development of error mitigation schemes without having to rely on physical hardware, and lastly, the identification of circuit optimizations. Different types of quantum circuits and applications require specialized simulators for each task. To this end, MQT Qudits offers two simulation options, each one with its merits and limitations. Again, we will examine both from the user and technical perspectives.

13.4.1 The User's Perspective

MQT Qudits offers a user-friendly interface that hides the complexity of simulating quantum circuits. In Figure 13.4, a simple demonstration of how to use the framework for quantum circuit simulations is shown. There are two levels of complexity for performing a simulation depending on the required level of customization. Once a quantum circuit is defined, you can call the simulate method of the circuit instance, which will return a job object upon completion of the execution. It is possible to retrieve the quantum state from the job, by calling "`get_state_vector()`". Alternatively, it is possible to choose a backend, thanks to the method "`get_backend(backend_name)`", between a tensor network simulator ("tnsim"), a mixed-dimensional decision diagram simulator ("misim", [10]), and a suite of simple emulators representing physical devices, again by using "`get_backend(fake_backend_name)`". These latter, prefixed with "fake", facilitate compilation and noisy simulation of quantum circuits with device-specific properties. Upon backend selection, the environment is ready to run the simulation of the circuit.

A particularly relevant feature of a quantum circuit simulator is noise-aware simulation, which allows for estimating the quality and feasibility of execution on a given quantum hardware. Figure 13.5 shows how to construct a noise model by specifying probabilities for generalized Pauli errors, that are described as $X^{\alpha} \cdot Z^{\beta}$, where X and Z flips are analogous to the qubit case, extended to the qudit Hilbert space of dimension d, while alfa and beta take value from 0 to $d-1$. Two novelties are the introduction of the subspace depolarizing noise, that is modelled like in the qubit case but adapted to act only on a subpace spanned by two dimensions at a time–inspired by empirical results [74], and the introduction of the dephasing noise that is described as the phase flip on a specific qudit dimension. The framework allows for fine-grained control over noise channels, supporting noise on local, as well as, multi-qudit operations with different effects on target and control qudits, and various other noise channels as detailed in the documentation. This approach enables the simulation of realistic noise processes in higher-dimensional quantum systems, crucial for assessing the performance of qudit-based quantum algorithms and error correction schemes. Different types of noise instances can be combined in a noise model and input as parameters of a simulation. The knowledge about the noise of a system can be easily integrated by adding it as an option to the simulation or by choosing a "fake" backend that will possess this knowledge by default.

```python
# import components and circuit
from mqt.qudits.simulation import MQTQuditProvider

# program written before
state = circuit.simulate()

# alternatively
provider = MQTQuditProvider()
backend = provider.get_backend("tnsim")
job = backend.run(circuit)
result = job.result()
state = result.get_state_vector()
```

Figure 13.4 Simulation of a qudit circuit with MQT Qudits' TnSim.

13.4.2 The Technical Perspective

The simplicity of use of the simulation components of MQT Qudits is due to the implementation of a modular architecture. The entry point is a provider object that serves as an abstraction layer for various backend implementations. For brevity, this section will cover only the most novel backends of the library: MiSiM [10] and TnSim. MiSiM, a C++ implementation, extends edge-weighted *Decision Diagrams* (DDs, [101], [102]), or QMDDs [100], to efficiently simulate mixed-dimensional qudit systems. By dynamically capturing qudit dimensionality, MiSiM significantly reduces memory requirements, enabling the simulation of complex circuits, e.g., exceeding, in some cases, 100 qutrits. This DD-based approach exploits redundancies and symmetries in the representation of quantum states and operations, allowing direct manipulation of operations on the diagram structure.

Additionally, DDs can efficiently represent sparse data, becoming a powerful tool for simulating circuits with non-trivial quantum correlations by representing states that would otherwise require exponential resources. TnSim, conversely, utilizes *tensor networks* representations via Google's TensorNetwork API [218] in Python. The tensor network simulator contained in MQT Qudit takes care of the construction of the network and the dynamical allocation of nodes in it and leverages optimized tensor contraction algorithms, offering performance advantages for circuits with specific entanglement structures. Tensor networks efficiently represent quantum many-body states with limited entanglement, particularly those obeying area-law scaling [109]. They decompose high-dimensional states into contracted lower-dimensional tensors, reducing computational complexity from exponential to polynomial for many physical systems.

Both backends support noise-aware simulation of quantum circuits through a stochastic approach [150], [151], [153], that we also described in *Chapter* 5. This method appends qudit Pauli gates after each operation with a specific probability described in the noise instance (Figure 13.5). For gates operating in subspaces, such as local subspace rotations [9], the noise is applied as subspace X, Y or Z operations, on the subspaces listed in the instance; if not, the dimensions selected are those of the rotation. The noise model in Fig. 13.5 encodes the logic determining whether each gate has a noise operation appended and, for entangling operations, whether noise is applied to the target, control(s), or a subset of qudits involved. A particularly simple, yet highly relevant, extension is the integration of numerical distortions of the gate parameters in the circuit associated with a specific noise channel, or the inclusion of subspace error operations for all types of gates.

This stochastic noise simulation method efficiently models decoherence and gate errors in quantum systems without the need for full density-matrix calculations. However, it requires multiple simulation runs to create a statistical distribution of results, which fluctuates due to operation-specific noise. Parallel processing could conceivably mitigate this limitation by executing runs simultaneously on several processes.

In summary, the library provides a suite of complementary simulators that leverage two data structures and different implementations. These can provide solutions with varying performance depending on the use case, the complexity of the noise to simulate, and the user's preference. Users can choose the most suitable simulator based on their specific requirements and computational constraints.

13.5 Quantum Circuit Compilation

Part IV describes how compilers can be useful in automating the transformation of abstract quantum algorithms into executable sequences for specific hardware, bridging theory, and implementation. The MQT Qudits compiler integrates quantum control theory, optimization, and custom heuristics (which have been discussed in Section 2.3.4, *Chapter* 10 and *Chapter* 11) to address coherence and fidelity challenges. It encompasses gate decomposition, mapping, and noise adaptation, which is crucial to realize the quantum algorithmic potential in near-term qudit processors. The compiler aims to be accessible to a wide range of users, from novices exploring mixed-dimensional quantum computing to well-resourced laboratories, while offering flexible, modular, and computationally efficient software that balances user-friendly operation with advanced technical capabilities. Therefore, it enables users to exploit these systems without the need to manually design circuits for every application–a process that would otherwise be error-prone and highly specific to the hardware. This section illustrates the use of the compiler.

13.5.1 The User Perspective

Figure 13.6 outlines how the user can exploit MQT Qudits for compilation tasks. The code shows how to set the initial state of the quantum circuit as a first operation, starting from the numpy array [195] of the state. After setting the initial state (line 8), the user can define the rest of the quantum circuit and directly compile the circuit for a target device (lines 11-20), just by stating the name of the device, prefixed by "fake". In this case, the compilation chooses some default flags and returns the compiled circuit compatible with the device to be executed on. Although highly flexible, the compiler is constantly evolving. The different suffices (O0, O1, and O2) declare the degree of optimization of the quantum circuits; in the O1 case it is possible to choose the optimization technique between "resynth" and "adapt". Multi-body gate compilation will be supported in future releases, while the transpilation of multi-controlled rotations is already implemented. For users interested in benchmarking, tuning the compilation process, and integrating their own passes into the compilation of a quantum circuit (lines 23-30), the required code is minimal and involves only three steps:

1. instantiation of the qudit compiler,
2. selection of compilation steps, and
3. choice of a target backend.

In this case, the compiler is returning the compiled circuit, which can then be either simulated or passed to the API of the actual device, after using the correct backend, e.g., "innsbruck01". The compiler enhancements outlined in this section significantly advance both its user-friendliness and efficiency. These developments are currently being refined in a distinct branch ("compiler_int") in preparation for a detailed quality evaluation.

13.5.2 The Technical Perspective

Shielding users from the complexity of the compilation process necessitates the development of sophisticated methods and implementations. The compiler is designed as a modular, pass-based tool that

prioritizes algorithmic efficiency, despite being entirely written in Python. In the following, we briefly review the main methods currently implemented in the MQT Qudits compiler, namely a state preparation routine, a single-qudit operations compiler, and a two-qudit operations compiler. The state preparation routine takes a quantum state in the form of a numpy array and outputs a sequence of local and controlled two-level rotation, with an increasing number of controls. The state preparation makes use of mixed-dimensional decision diagrams [10] implemented in Python for efficiently representing the state. A synthesis algorithm reads the DD structure, possibly approximates it if required by the user, and returns a sequence of quantum operations to generate the state. The execution of the multi-controlled operations is allowed by an implicit decomposition in qudit "CNOT" by using ancilla levels in the qudits [74]. We refer to the paper [12] for further details on the synthesis.

The compilation steps enabled in the process depend on the user's choice of passes. Depending on the selected passes, the compilation can target either single-qudit or two-qudit unitaries. The pass nomenclature indicates whether the compilation produces a physically executable sequence compatible with the backend ("Phys") or a logical sequence ("Log"). "Loc" denotes single-qudit unitary compilation, while "Ent" signifies two-qudit unitary compilation.

Single-qudit unitaries are compiled using sequences of 2-level subspace rotations. This reconstruction method aligns with the energy level graph of the individual qudit. The energy level graph [14] is a data structure that represents the internal structure of the qudit, storing information about the quality of rotations between two levels and tracking various properties of the decomposition on the physical system. This graph structure corresponds to the coupling map between the various qudit levels. The use of the energy level graph and graph optimization heuristics has led to improvements in the compilation of single-qudit unitaries, particularly in reducing the number of rotations required, especially phase rotations [14].

MQT Qudits compiles two-qudit unitaries of arbitrary dimensionality into sequences of controlled rotations (crot) and partial swap operations (pswap). These operations are then either directly implemented in hardware or further decomposed into a sequence of local operations and controlled-exchange gates, which is a qudit-embedded 2-level CNOT gate. This methodology [13] provides a computationally efficient means of compiling arbitrary interactions into standardized entangling gates, independent of the underlying hardware architecture. The compiler allows for variational compilation of the standardized entangling gates into the hardware-native gate set, tailoring the circuit to the specific hardware. Further extensions should include tools for compressing the generated circuits to further increase the efficiency. Controlled rotations are simply transpiled to match the mapping of the logic states on the architectures.

An additional standalone tool is the qubit compression mapper [11], a utility that recommends dimensionality adjustments in mixed-dimensional architectures to efficiently encode qubit circuits into qudit circuits. This tool facilitates the translation between different quantum information encoding schemes, potentially enabling more efficient use of available quantum resources.

13.6 Conclusion

MQT Qudits provides an open-source framework for mixed-dimensional quantum computing, including flexible automated tools for the design, specification, simulation, and compilation of mixed-dimensional quantum circuits. Considering the fundamental differences between qubit and qudit quantum computing, the framework is designed to be easily extendable to new kinds of gate operations that are expected to be developed as the field evolves. Similarly, the development and inclusion of more hardware-specific noise models for the various types of qudit quantum computing hardware is a point of great interest to enable accurate circuit simulation.

A particularly interesting topic is the dynamic evolution of the data structures during the simulation with the goal of tracking the usage of ancilla levels in the qudit. Finally, the compiler framework enables flexible and adaptive compilation, yet does not always achieve optimal circuits. Future efforts should prioritize automated multi-qudit gate compilation, and optimizing two-qudit gate compilations to minimize the resulting circuit complexity. Efforts should also include resynthesis procedures to reduce the final circuit depth. In order to facilitate these developments, MQT Qudits will, in the future, include scalable verification methods for mixed-dimensional systems.

This chapter served as the culmination of this Ph.D. thesis, which has laid the foundation for expressing the potential of mixed-dimensional quantum computing. By addressing the challenges in simulation and compilation, this thesis provides innovative methodologies and tools to support the design and execution of quantum algorithms for systems beyond qubits. The results, integrated into an open-source framework, shed light on the way for a new paradigm of quantum computation, combining theoretical rigor with practical utility and enabling scalable and reliable applications on mixed-dimensional quantum architectures.

```python
# import components and circuit
from mqt.qudits.simulation import MQTQuditProvider
from mqt.qudits.simulation.noise_tools.noise import \
    Noise, NoiseModel

basic_error = Noise(probability_depolarizing=0.005, \
                    probability_dephasing=0.005)
subspace_dynamic_error = SubspaceNoise(
    probability_depolarizing=1e-4,  \
    probability_dephasing=1e-4, levels=[]
)

suberror_01 = SubspaceNoise(
        probability_depolarizing=0.005, \
        probability_dephasing=0.005, levels=(0, 1))
suberror_01_cex = SubspaceNoise(
        probability_depolarizing=0.010, \
        probability_dephasing=0.010, levels=(0, 1)
)

# Add errors to noise_tools model
noise_model = NoiseModel()

# Gates on higher abstraction
noise_model.add_nonlocal_quantum_error(
        basic_error, ["csum", "ls"])
noise_model.add_quantum_error_locally(
        basic_error, ["cuone", "cutwo", "cumulti", \
                    "h", "perm", "rdu", "s", "x", "z"]
)

# Physical gates
noise_model.add_nonlocal_quantum_error(
            suberror_01, ["ms"])
noise_model.add_nonlocal_quantum_error(
            suberror_01_cex, ["cx"])
noise_model.add_quantum_error_locally(
            subspace_dynamic_error, ["rxy", "rh"])
noise_model.add_quantum_error_locally(
            subspace_dynamic_error, ["rz"])

provider = MQTQuditProvider()
backend = provider.get_backend("tnsim")
job = backend.run(circuit, noise_model=noise_model)
result = job.result()
counts = result.get_counts()
```

Figure 13.5 Simulation of a qudit circuit with a tailored noise model through MQT Qudits.

```
# import components and circuit
from mqt.qudits.simulation import MQTQuditProvider
from mqt.qudits.compiler import QuditCompiler
provider = MQTQuditProvider()

# optionally the user can set the initial state
state = np.array(...)
circuit.set_initial_state(state)
# rest of the program
# simple interface
new_circuit = circuit.compileO0("faketraps2six")

# or
new_circuit = circuit.compileO1("faketraps2six", \
                                "adapt")
# or
new_circuit = circuit.compileO1("faketraps2six", \
                                "resynth")
# or
new_circuit = circuit.compileO2("faketraps2six")

# Alternatively for the experts
qudit_compiler = QuditCompiler()

# choice of the passes
passes = ["PhyLocQRPass", "PhyEntQRCEXPass",
        "PhyEntSimplePass", "PhyMultiSimplePass"]

device = provider.get_backend("faketraps2six")
new_circuit = qudit_compiler.compile(device, \
                        circuit, passes)
```

Figure 13.6 A prototypical compilation of a quantum circuit on MQT Qudits.

14 Conclusions and Outlook

Quantum computing has made significant progress in terms of hardware, theory and quantum software development. Despite the advancements in qubit-based technology, involving qudits for calculations might play a key role in overcoming not only the limitations of classical machines but also those of current quantum devices. This new native synergy mixes the best capabilities of both worlds, offering significant potential for tailored methods in error mitigation, optimal circuit, and application designs. This new paradigm of quantum computing, called *mixed-dimensional quantum computing*, sadly, has a scarcity of automated software solutions, which enforces that a considerable portion of the design process in quantum computing still heavily relies on manual techniques. Quantum software for mixed-dimensional quantum computating faces design challenges reminiscent of those previously encountered in classical computing, therefore, the envisioned quantum software flow should parallel that of classical software.

This Ph.D. thesis laid the groundwork for developing methods and state-of-the-art quantum software that will unleash the potential of mixed-dimensional quantum computing by addressing two specific challenges in design software: simulation and compilation. Specifically, this involves using efficient and novel methodologies inspired by fundamental techniques previously developed in traditional circuits and systems, such as data structures (e.g., adapted decision diagrams) and methods like heuristic search, alongside quantum information.

Classical simulation of quantum circuits is crucial for various aspects of quantum application development, as it allows for algorithm verification when access to appropriate hardware is restricted, for assessing quantum hardware performance, and for validating the corresponding results. Despite its utility, simulating quantum systems with mixed dimensions poses specific difficulties due to the exponential expansion of the state space as both the number of qudits and their individual dimensions increase, rendering traditional qubit simulation techniques insufficient right off the bat. In this way, the use of specialized data structures improves performance and memory consumption, and the introduction of noise models validates the results of mixed-dimensional quantum computers.

Two different contributions have been proposed to enable the simulation of mixed-dimensional quantum circuits. First, the introduction of mixed-dimensional decision diagrams, a data-structure inspired by classical design automation, were adapted to accelerate the simulation of mixed-dimensional circuits on classical machines. Second, the formalization of the first new noise model for mixed-dimensional systems, accompanied by a Monte-Carlo simulation framework, designed with an efficient exploitation of data-structures in mind. These approaches significantly improve the current state of the art in classical quantum circuit simulation for mixed-dimensional systems and enhance the understanding of realistic applications for such architectures.

This Ph.D. thesis addressed the challenge of translating high-level quantum algorithms into executable operations on mixed-dimensional hardware, namely quantum compilation, aiming to find the shortest possible sequence of quantum operations needed to implement a given quantum state or operation.

Several methods were proposed to address this challenge, including compressing qubit circuits into qudit architectures, preparing arbitrary quantum states using decision diagrams, and compiling local and entangling gates for high-dimensional systems. These approaches automate compilation for mixed-dimensional systems for the first time. The compilation process considered varying dimensions, diverse

types of operations, and hardware-specific constraints while striving to optimize the depth of the circuit, its resistance to errors, as well as runtime and memory consumption.

Central to the practical realization of these advances is the development of *MQT Qudits*, a part of *Munich Quantum Toolkit*. This software framework supports mixed-dimensional quantum computing by integrating simulation and compilation capabilities into a unified Python-based interface and language.

The contributions of this thesis demonstrate that the integration of classical systems' automation techniques into quantum software can improve the simulation and compilation of quantum circuits, taking mixed-dimensional quantum computing from academic theory to a fully operational system. Moreover, the results presented in this thesis emphasize the critical role of quantum software in enabling quantum algorithms to run efficiently, reliably, and scalably on mixed-dimensional quantum computers.

In conclusion, this Ph.D. thesis highlights the importance of developing automated software tools for mixed-dimensional quantum computing and, by addressing key challenges with open-source solutions, advances the field while laying a foundation for its next generation and emphasizing the transformative role of classical computing design methods in shaping its future.

References

[1] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*, 10th. New York, USA: Cambridge University Press, 2011.

[2] R. P. Feynman, "Simulating physics with computers", *International Journal of Theoretical Physics*, vol. 21, no. 6–7, pp. 467–488, 1982.

[3] H. Abraham, AduOffei, *et al.*, *Qiskit: An open-source framework for quantum computing*, 2021.

[4] R. Wille, L. Berent, *et al.*, *The MQT Handbook: A Summary of Design Automation Tools and Software for Quantum Computing*, 2024. arXiv: 2405.17543 [quant-ph].

[5] G. Quantum AI Team, *Cirq: A python framework for creating, editing, and invoking noisy intermediate scale quantum (nisq) circuits*, https://github.com/quantumlib/Cirq, 2024.

[6] Microsoft, *Quantum development kit*, https://microsoft.com/en-us/quantum/development-kit, 2024.

[7] The CUDA-Q development team, *CUDA-Q*, https://github.com/NVIDIA/cuda-quantum.

[8] V. Bergholm, J. Izaac, *et al.*, *Pennylane: Automatic differentiation of hybrid quantum-classical computations*, 2022. arXiv: 1811.04968 [quant-ph].

[9] M. Ringbauer, M. Meth, *et al.*, "A universal qudit quantum processor with trapped ions", *Nature Physics*, vol. 18, no. 9, pp. 1053–1057, 2022.

[10] K. Mato, S. Hillmich, and R. Wille, "Mixed-dimensional quantum circuit simulation with decision diagrams", in *2023 IEEE International Conference on Quantum Computing and Engineering (QCE)*, vol. 01, 2023, pp. 978–989.

[11] K. Mato, S. Hillmich, and R. Wille, "Compression of qubit circuits: Mapping to mixed-dimensional quantum systems", in *International Conference on Quantum Software (QSW)*, 2023, pp. 155–161.

[12] K. Mato, S. Hillmich, and R. Wille, "Mixed-dimensional qudit state preparation using edge-weighted decision diagrams", 2024.

[13] K. Mato, M. Ringbauer, S. Hillmich, and R. Wille, "Compilation of entangling gates for high-dimensional quantum systems", in *Asia and South Pacific Design Automation Conf.*, ser. ASPDAC '23, Tokyo, Japan, 2023, pp. 202–208.

[14] K. Mato, M. Ringbauer, S. Hillmich, and R. Wille, "Adaptive compilation of multi-level quantum operations", in *International Conference on Quantum Computing and Engineering (QCE)*, 2022, pp. 484–491.

[15] K. Mato, M. Ringbauer, L. Burgholzer, and R. Wille, *MQT Qudits: A Software Framework for Mixed-Dimensional Quantum Computing*, 2024. arXiv: 2410.02854 [quant-ph].

[16] Y. Wang, Z. Hu, B. C. Sanders, and S. Kais, "Qudits and high-dimensional quantum computing", *Front. Phys.*, vol. 8, 2020.

[17] J. J. Sakurai and J. Napolitano, *Modern Quantum Mechanics*, en, 2nd ed. Cambridge, England: Cambridge University Press, 2017.

[18] J. Preskill, “Quantum computing in the NISQ era and beyond”, *Quantum*, vol. 2, p. 79, 2018.

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

[20] I. Pogorelov, T. Feldker, *et al.*, “Compact Ion-Trap Quantum Computing Demonstrator”, *PRX Quantum*, vol. 2, p. 020 343, 2021.

[21] F. Flamini, N. Spagnolo, and F. Sciarrino, “Photonic quantum information processing: A review”, *Reports Prog. Phys.*, vol. 82, p. 016 001, 2019.

[22] B. P. Lanyon, M. Barbieri, *et al.*, “Simplifying quantum logic using higher-dimensional Hilbert spaces”, *Nat. Phys.*, vol. 5, no. 2, pp. 134–140, 2008.

[23] Z. Gedik, I. A. Silva, *et al.*, “Computational speed-up with a single qudit”, *Sci. Rep.*, vol. 5, p. 14 671, 2015.

[24] X. Zhan, J. Li, *et al.*, “Linear optical demonstration of quantum speed-up with a single qudit”, *Optics Express*, vol. 23, p. 18 422, 14 2015.

[25] X. Zhang, M. Um, *et al.*, “State-independent experimental test of quantum contextuality with a single trapped ion”, *Phys. Rev. Lett.*, vol. 110, p. 070 401, 2013.

[26] M. Ringbauer, T. R. Bromley, *et al.*, “Certification and quantification of multilevel quantum coherence”, *Phys. Rev. X*, vol. 8, p. 041 007, 2018.

[27] X.-M. Hu, Y. Guo, *et al.*, “Beating the channel capacity limit for superdense coding with entangled ququarts”, *Sci. Adv.*, vol. 4, no. 7, eaat9304, 2018.

[28] M. Malik, M. Erhard, *et al.*, “Multi-photon entanglement in high dimensions”, *Nature Photonics*, vol. 10, pp. 248–252, 2016.

[29] M. Kononenko, M. Yurtalan, *et al.*, “Characterization of control in a superconducting qutrit using randomized benchmarking”, *Phys. Rev. Res.*, vol. 3, no. 4, p. L042007, 2021.

[30] A. Morvan, V. V. Ramasesh, *et al.*, “Qutrit randomized benchmarking”, *Phys. Rev. Letters*, vol. 126, p. 210 504, 2021.

[31] J. Ahn, T. Weinacht, and P. Bucksbaum, “Information storage and retrieval through quantum phase”, *Science*, vol. 287, no. 5452, pp. 463–465, 2000.

[32] J. R. Weggemans, A. Urech, *et al.*, “Solving correlation clustering with QAOA and a Rydberg qudit system: a full-stack approach”, 2021. arXiv: 2106.11672.

[33] C. Godfrin, A. Ferhat, *et al.*, “Operating quantum states in single magnetic molecules: Implementation of Grover’s quantum algorithm”, *Phys. Rev. Letters*, vol. 119, p. 187 702, 2017.

[34] B. E. Anderson, H. Sosa-Martinez, *et al.*, “Accurate and robust unitary transformations of a high-dimensional quantum system”, *Phys. Rev. Letters*, vol. 114, p. 240 401, 2015.

[35] V. Kasper, D. González-Cuadra, *et al.*, “Universal quantum computation and quantum error correction with ultracold atomic mixtures”, 2020. arXiv: 2010.15923.

[36] F. Tacchino, A. Chiesa, *et al.*, “A proposal for using molecular spin qudits as quantum simulators of light–matter interactions”, *J. Mater. Chem. C*, vol. 9, pp. 10 266–10 275, 32 2021.

[37] Y. Chi, J. Huang, *et al.*, “A programmable qudit-based quantum processor”, *Nature Communications*, vol. 13, p. 1166, 1 2022.

[38] D. Gottesman, *The heisenberg representation of quantum computers*, 1998. arXiv: quant-ph/9807006 [quant-ph].

[39] S. Aaronson and D. Gottesman, “Improved simulation of stabilizer circuits”, *Phys. Rev. A*, vol. 70, p. 052 328, 5 2004.

[40] D. Gottesman, “Theory of fault-tolerant quantum computation”, *Phys. Rev. A*, vol. 57, pp. 127–137, 1 1998.

[41] R. Sarkar and T. J. Yoder, “The qudit pauli group: Non-commuting pairs, non-commuting sets, and structure theorems”, *Quantum*, vol. 8, p. 1307, 2024.

[42] G. K. Brennen, S. S. Bullock, and D. P. O’Leary, “Efficient circuits for exact-universal computation with qudits”, *Quantum Info. Comput.*, vol. 6, no. 4, pp. 436–454, 2006.

[43] P. Hrmo, B. Wilhelm, *et al.*, “Native qudit entanglement in a trapped ion quantum processor”, *Nature Communications*, vol. 14, no. 1, p. 2242, 2023.

[44] D. Aharonov, *A simple proof that toffoli and hadamard are quantum universal*, 2003. arXiv: quant-ph/0301040 [quant-ph].

[45] A. Sawicki and K. Karnas, “Universality of single-qudit gates”, *Annales Henri Poincaré*, vol. 18, no. 11, pp. 3515–3552, 2017.

[46] A. Sawicki, L. Mattioli, and Z. Zimborás, “Universality verification for a set of quantum gates”, *Phys. Rev. A*, vol. 105, p. 052 602, 5 2022.

[47] C. Dankert, R. Cleve, J. Emerson, and E. Livine, “Exact and approximate unitary 2-designs and their application to fidelity estimation”, *Phys. Rev. A*, vol. 80, no. 1, 2009.

[48] *Quantum t-designs: t-wise independence in the quantum world, author=Andris Ambainis and Joseph Emerson*, 2007. arXiv: quant-ph/0701126 [quant-ph].

[49] Y. Nakata, D. Zhao, *et al.*, “Quantum Circuits for Exact Unitary t-Designs and Applications to Higher-Order Randomized Benchmarking”, *PRX Quantum*, vol. 2, no. 3, 2021.

[50] P. W. Shor, “Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer”, *SIAM Journal on Computing*, vol. 26, no. 5, pp. 1484–1509, 1997.

[51] L. K. Grover, “A fast quantum mechanical algorithm for database search”, G. L. Miller, Ed., ACM, 1996, pp. 212–219.

[52] Y. Cao, J. Romero, *et al.*, “Quantum chemistry in the age of quantum computing”, *Chemical Reviews*, vol. 119, no. 19, pp. 10 856–10 915, 2019.

[53] B. Lanyon, M. Barbieri, *et al.*, “Quantum computing using shortcuts through higher dimensions”, *Nature Physics*, vol. 5, 2008.

[54] D. A. Drozhzhin, A. S. Nikolaeva, E. O. Kiktenko, and A. K. Fedorov, “Transpiling quantum assembly language circuits to a qudit form”, *Entropy*, vol. 26, no. 12, p. 1129, 2024.

[55] A. S. Nikolaeva, E. O. Kiktenko, and A. K. Fedorov, “Generalized toffoli gate decomposition using ququints: Towards realizing grover’s algorithm with qudits”, *Entropy*, vol. 25, no. 2, 2023.

[56] A. S. Nikolaeva, E. O. Kiktenko, and A. K. Fedorov, *Efficient realization of quantum algorithms with qudits*, 2021.

[57] A. S. Nikolaeva, E. O. Kiktenko, and A. K. Fedorov, “Universal quantum computing with qubits embedded in trapped-ion qudits”, *Physical Review A*, vol. 109, no. 2, 2024.

[58] S. Das and F. Caruso, “A hybrid-qudit representation of digital rgb images”, 2022.

[59] N. L. Wach, M. S. Rudolph, F. Jendrzejewski, and S. Schmitt, “Data re-uploading with a single qudit”, *Quantum Machine Intelligence*, vol. 5, no. 2, 2023.

[60] S. Roca-Jerat, J. Román-Roche, and D. Zueco, "Qudit machine learning", *Machine Learning: Science and Technology*, vol. 5, no. 1, p. 015 057, 2024.

[61] A. Bottarelli, S. Schmitt, and P. Hauke, *Inequality constraints in variational quantum circuits with qudits*, 2024. arXiv: 2410.07674 [quant-ph].

[62] L. Ekstrom, H. Wang, and S. Schmitt, *Variational quantum multi-objective optimization*, 2024. arXiv: 2312.14151 [quant-ph].

[63] M. G. De Andoin, A. Bottarelli, *et al.*, "Formulation of the electric vehicle charging and routing problem for a hybrid quantum-classical search space reduction heuristic", in *2023 IEEE 26th International Conference on Intelligent Transportation Systems (ITSC)*, IEEE, 2023, pp. 5318–5323.

[64] F. D. M. Haldane, "Nonlinear Field Theory of Large-Spin Heisenberg Antiferromagnets: Semiclassically Quantized Solitons of the One-Dimensional Easy-Axis Néel State", *Physical Review Letters*, vol. 50, no. 15, 1983.

[65] C. Senko, P. Richerme, *et al.*, "Realization of a Quantum Integer-Spin Chain with Controllable Interactions", *Phys. Rev. X*, vol. 5, p. 021 026, 2015.

[66] C. L. Edmunds, E. Rico, *et al.*, *Constructing the spin-1 Haldane phase on a qudit quantum processor*, 2024.

[67] Y. Deller, S. Schmitt, *et al.*, "Quantum approximate optimization algorithm for qudit systems", 2023.

[68] D. González-Cuadra, T. V. Zache, *et al.*, "Hardware Efficient Quantum Simulation of Non-Abelian Gauge Theories with Qudits on Rydberg Platforms", *Phys. Rev. Lett.*, vol. 129, p. 160 501, 16 2022.

[69] E. J. Gustafson, "Prospects for simulating a qudit-based model of $(1+1)\mathrm{D}$ scalar qed", *Phys. Rev. D*, vol. 103, p. 114 505, 11 2021.

[70] D. M. Kurkcuoglu, M. S. Alam, *et al.*, "Quantum simulation of ϕ^4 theories in qudit systems", *TBD*, 2021.

[71] L. Lamata, A. Mezzacapo, J. Casanova, and E. Solano, "Efficient quantum simulation of fermionic and bosonic models in trapped ions", *EPJ Quantum Technology*, vol. 1, p. 9, 2014.

[72] J. Casanova, L. Lamata, *et al.*, "Quantum simulation of quantum field theories in trapped ions", *Phys. Rev. Lett.*, vol. 107, p. 260 501, 26 2011.

[73] T. Navickas, R. J. MacDonell, *et al.*, *Experimental Quantum Simulation of Chemical Dynamics*, 2024.

[74] M. Meth, J. F. Haase, *et al.*, *Simulating 2D lattice gauge theories on a qudit quantum computer*, 2024. arXiv: 2310.12110 [quant-ph].

[75] G. Calajò, G. Magnifico, *et al.*, *Digital Quantum Simulation of a (1+1)D SU(2) Lattice Gauge Theory with Ion Qudits*, 2024.

[76] P. P. Popov, V. Kasper, *et al.*, "Non-perturbative signatures of fractons in the twisted multi-flavor Schwinger Model", *Preprint at arXiv:2405.00745*, 2024.

[77] A. Litteken, J. M. Baker, and F. T. Chong, "Communication trade offs in intermediate qudit circuits", in *2022 IEEE 52nd International Symposium on Multiple-Valued Logic (ISMVL)*, 2022, pp. 43–49.

[78] L. Seifert, J. Chadwick, *et al.*, "Time-efficient qudit gates through incremental pulse re-seeding", 2022, pp. 304–313.

[79] S. A. Moses, C. H. Baldwin, *et al.*, "A race-track trapped-ion quantum processor", *Phys. Rev. X*, vol. 13, p. 041 052, 4 2023.

[80] W. K. Hensinger, "Quantum computer based on shuttling trapped ions", *Nature*, vol. 592, no. 7853, pp. 190–191, 2021.

[81] M.-A. Lemonde, D. Lachance-Quirion, *et al.*, *Hardware-efficient fault tolerant quantum computing with bosonic grid states in superconducting circuits*, 2024. arXiv: 2409.05813 [quant-ph].

[82] J. Guillaud, J. Cohen, and M. Mirrahimi, "Quantum computation with cat qubits", *SciPost Physics Lecture Notes*, 2023.

[83] R. Acharya, L. Aghababaie-Beni, *et al.*, *Quantum error correction below the surface code threshold*, 2024. arXiv: 2408.13687 [quant-ph].

[84] L. Postler, F. Butt, *et al.*, "Demonstration of fault-tolerant steane quantum error correction", *PRX Quantum*, vol. 5, p. 030 326, 3 2024.

[85] M. P. da Silva, C. Ryan-Anderson, *et al.*, *Demonstration of logical qubits and repeated error correction with better-than-physical error rates*, 2024. arXiv: 2404.02280 [quant-ph].

[86] D. Bluvstein, S. J. Evered, *et al.*, "Logical quantum processor based on reconfigurable atom arrays", *Nature*, vol. 626, no. 7997, pp. 58–65, 2023.

[87] A. deMarti iOlius, P. Fuentes, *et al.*, "Decoding algorithms for surface codes", *Quantum*, vol. 8, p. 1498, 2024.

[88] B. Barber, K. M. Barnes, *et al.*, *A real-time, scalable, fast and highly resource efficient decoder for a quantum computer*, 2024. arXiv: 2309.05558 [quant-ph].

[89] R. E. Bryant, "Graph-based algorithms for boolean function manipulation", *Computers, IEEE Transactions on*, vol. 100, no. 8, pp. 677–691, 1986.

[90] R. E. Bryant, "Symbolic boolean manipulation with ordered binary-decision diagrams", *ACM Computing Surveys (CSUR)*, vol. 24, no. 3, pp. 293–318, 1992.

[91] I. Wegener, *Branching programs and binary decision diagrams: theory and applications*. SIAM, 2000.

[92] J. Gergov and C. Meinel, "Efficient boolean manipulation with obdd's can be extended to fbdd's", *IEEE Transactions on Computers*, vol. 43, no. 10, pp. 1197–1209, 1994.

[93] R. Drechsler, A. Sarabi, *et al.*, "Efficient representation and manipulation of switching functions based on ordered kronecker functional decision diagrams", in *Proceedings of the 31st annual Design Automation Conference*, 1994, pp. 415–419.

[94] R. I. Bahar, E. A. Frohm, *et al.*, "Algebric decision diagrams and their applications", *Formal methods in system design*, vol. 10, pp. 171–206, 1997.

[95] S.-i. Minato, "Zero-suppressed bdds for set manipulation in combinatorial problems", in *Proceedings of the 30th International Design Automation Conference*, 1993, pp. 272–277.

[96] L. Vinkhuijzen, T. Coopmans, *et al.*, "Limdd: A decision diagram for simulation of quantum computing including stabilizer states", *Quantum*, vol. 7, p. 1108, 2023.

[97] G. F. Viamontes, I. L. Markov, and J. P. Hayes, "Improving gate-level simulation of quantum circuits", *Quantum Information Processing*, vol. 2, pp. 347–380, 2003.

[98] D. Miller and M. Thornton, "QMDD: A decision diagram structure for reversible and quantum circuits", IEEE, 2006.

[99] S. Wang, C. Lu, I. Tsai, and S. Kuo, "An XQDD-based verification method for quantum circuits", *IEICE Trans. Fundam. Electron. Commun. Comput. Sci.*, vol. 91-A, no. 2, pp. 584–594, 2008.

[100] P. Niemann, R. Wille, *et al.*, "QMDDs: Efficient quantum function representation and manipulation", *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.*, vol. 35, no. 1, pp. 86–99, 2016.

[101] A. Zulehner, S. Hillmich, and R. Wille, “How to efficiently handle complex values? Implementing decision diagrams for quantum computing”, ACM, 2019, pp. 1–7.

[102] A. Zulehner and R. Wille, “Advanced simulation of quantum computations”, vol. 38, no. 5, pp. 848–859, 2019.

[103] S. Hillmich, I. L. Markov, and R. Wille, “Just like the real thing: Fast weak simulation of quantum computation”, IEEE, 2020, pp. 1–6.

[104] P. Niemann, R. Wille, and R. Drechsler, “Efficient synthesis of quantum circuits implementing clifford group operations”, in *2014 19th Asia and South Pacific Design Automation Conference (ASP-DAC)*, IEEE, 2014, pp. 483–488.

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

[106] M. Soeken, R. Wille, *et al.*, “Synthesis of reversible circuits with minimal lines for large functions”, in *17th Asia and South Pacific Design Automation Conference*, IEEE, 2012, pp. 85–92.

[107] A. Zulehner and R. Wille, “One-pass design of reversible circuits: Combining embedding and synthesis for reversible logic”, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 5, pp. 996–1008, 2017.

[108] J. C. Bridgeman and C. T. Chubb, “Hand-waving and interpretive dance: An introductory course on tensor networks”, *Journal of Physics A: Mathematical and Theoretical*, vol. 50, no. 22, p. 223 001, 2017.

[109] R. Orús, “A practical introduction to tensor networks: Matrix product states and projected entangled pair states”, *Annals of Physics*, vol. 349, pp. 117–158, 2014.

[110] G. Evenbly and G. Vidal, “Tensor network states and geometry”, *Journal of Statistical Physics*, vol. 145, no. 4, pp. 891–918, 2011.

[111] J. I. Cirac and F. Verstraete, “Renormalization and tensor product states in spin chains and lattices”, *Journal of Physics A: Mathematical and Theoretical*, vol. 42, no. 50, p. 504 004, 2009.

[112] R. Orús, “Advances on tensor network theory: Symmetries, fermions, entanglement, and holography”, *The European Physical Journal B*, vol. 87, 2014.

[113] G. Evenbly and G. Vidal, “Tensor network renormalization yields the multiscale entanglement renormalization ansatz.”, *Physical review letters*, vol. 115 20, p. 200 401, 2015.

[114] D. H. Menzel, *Mathematical Physics* (Dover Books on Physics), 2nd ed. Mineola, NY: Dover Publications, 1961.

[115] L. Chi-Chung, P. Sadayappan, and R. Wenger, “On optimizing a class of multi-dimensional loops with reduction for parallel execution”, *Parallel Processing Letters*, vol. 07, no. 02, pp. 157–168, 1997.

[116] J. Gray and S. Kourtis, “Hyper-optimized tensor network contraction”, *Quantum*, vol. 5, p. 410, 2021.

[117] J. Gray and G. K.-L. Chan, “Hyperoptimized approximate contraction of tensor networks with arbitrary geometry”, *Physical Review X*, vol. 14, no. 1, 2024.

[118] Z. Y. Xie, H. J. Liao, *et al.*, “Optimized contraction scheme for tensor-network states”, *Physical Review B*, vol. 96, no. 4, 2017.

[119] M. Lubasch, J. I. Cirac, and M.-C. Bañuls, “Unifying projected entangled pair state contractions”, *New Journal of Physics*, vol. 16, no. 3, p. 033 014, 2014.

[120] A. Hashim, L. B. Nguyen, *et al.*, *A practical introduction to benchmarking and characterization of quantum computers*, 2024. arXiv: 2408.12064 [quant-ph].

[121] S. Chatterjee, *Distances between probability measures*, 2007.

[122] V. Vedral, “The role of relative entropy in quantum information theory”, *Reviews of Modern Physics*, vol. 74, no. 1, pp. 197–234, 2002.

[123] R. Jozsa, “Fidelity for mixed quantum states”, *Journal of Modern Optics*, vol. 41, no. 12, pp. 2315–2323, 1994.

[124] D. Aharonov, A. Kitaev, and N. Nisan, *Quantum circuits with mixed states*, 1998. arXiv: quant-ph/9806029 [quant-ph].

[125] K. Mayer and E. Knill, “Quantum process fidelity bounds from sets of input states”, *Physical Review A*, vol. 98, no. 5, 2018.

[126] A. Jamiołkowski, “Linear transformations which preserve trace and positive semidefiniteness of operators”, *Reports on Mathematical Physics*, vol. 3, no. 4, pp. 275–278, 1972.

[127] M.-D. Choi, “Completely positive linear maps on complex matrices”, *Linear Algebra and its Applications*, vol. 10, no. 3, pp. 285–290, 1975.

[128] A. Newell and H. A. Simon, “Computer science as empirical inquiry: Symbols and search”, *Commun. ACM*, vol. 19, no. 3, pp. 113–126, 1976.

[129] J. Pearl, *Heuristics: Intelligent search strategies for computer problem solving*. Addison-Wesley Pub. Co., Inc.,Reading, MA, 1984.

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

[131] T. J. Stavenger, E. Crane, *et al.*, “C2QA - bosonic qiskit”, 2022.

[132] B. Poór, Q. Wang, *et al.*, “Completeness for arbitrary finite dimensions of ZXW-calculus, a unifying calculus”, 2023.

[133] L. Burgholzer and R. Wille, “The power of simulation for equivalence checking in quantum computing”, IEEE, 2020, pp. 1–6.

[134] L. Burgholzer and R. Wille, “Advanced equivalence checking for quantum circuits”, vol. 40, no. 9, pp. 1810–1824, 2021.

[135] D. Volya and P. Mishra, “Quantum data compression for efficient generation of control pulses”, ACM, 2023, pp. 216–221.

[136] S. Jaques and T. Häner, “Leveraging state sparsity for more efficient quantum simulations”, vol. 3, no. 3, 2022.

[137] T. Jones, A. Brown, I. Bush, and S. C. Benjamin, “QuEST and high performance simulation of quantum computers”, *Sci. Rep.*, vol. 9, no. 1, 2019.

[138] S. Efthymiou, S. Ramos-Calderer, *et al.*, “Qibo: A framework for quantum simulation with hardware acceleration”, *Quantum Science and Technology*, vol. 7, no. 1, p. 015 018, 2021.

[139] X.-Z. Luo, J.-G. Liu, P. Zhang, and L. Wang, “Yao.jl: Extensible, efficient framework for quantum algorithm design”, *Quantum*, vol. 4, p. 341, 2020.

[140] S. V. Isakov, D. Kafri, *et al.*, *Simulations of quantum circuits with approximate noise using qsim and Cirq*, 2021. arXiv: 2111.02396.

[141] J. Hauschild and F. Pollmann, “Efficient numerical simulations with tensor networks: Tensor network python (TeNPy)”, *SciPost Phys. Lecture Notes*, 2018.

[142] J. Gray, “Quimb: A python package for quantum information and many-body calculations”, *Journal of Open Source Software*, vol. 3, no. 29, p. 819, 2018.

[143] M. Fishman, S. R. White, and E. M. Stoudenmire, “The ITensor Software Library for Tensor Network Calculations”, *SciPost Phys. Codebases*, p. 4, 2022.

[144] T. Chatterjee, A. Das, *et al.*, *QuDiet: A classical simulation platform for qubit-qudit hybrid quantum systems*, 2022. arXiv: 2211.07918.

[145] S. Hillmich, C. Hadfield, *et al.*, “Decision diagrams for quantum measurements with shallow circuits”, IEEE, 2021, pp. 24–34.

[146] R. Wille, S. Hillmich, and L. Burgholzer, “Decision diagrams for quantum computing”, in *Design Automation of Quantum Computers*. Springer, 2023, pp. 1–23.

[147] L. Yeh, “Scaling w state circuits in the qudit clifford hierarchy”, in *ACM Companion Proceedings of the 7th International Conference on the Art, Science, and Engineering of Programming*, 2023.

[148] D. M. Greenberger, M. A. Horne, and A. Zeilinger, “Going beyond bell’s theorem”, in *Bell’s Theorem, Quantum Theory and Conceptions of the Universe*, M. Kafatos, Ed. Springer, 1989, pp. 69–72.

[149] S. Rasmussen, K. Christensen, *et al.*, “Superconducting circuit companion—an introduction with worked examples”, *PRX Quantum*, vol. 2, no. 4, 2021.

[150] W. Berquist, D. Lykov, M. Liu, and Y. Alexeev, “Stochastic approach for simulating quantum noise using tensor networks”, in *Third International Workshop on Quantum Computing Software (QCS)*, 2022, pp. 107–113.

[151] T. Grurl, R. Kueng, J. Fuß, and R. Wille, “Stochastic quantum circuit simulation using decision diagrams”, in *Design, Automation & Test in Europe Conference*, pp. 194–199.

[152] T. Grurl, J. Fuß, and R. Wille, “Optimized density matrix representations : Improving the basis for noise-aware quantum circuit design tools”, in *2023 IEEE 53rd International Symposium on Multiple-Valued Logic (ISMVL)*, 2023, pp. 141–146.

[153] T. Grurl, J. Fuß, and R. Wille, “Noise-aware quantum circuit simulation with decision diagrams”, *Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 42, no. 3, pp. 860–873,

[154] B. Giles and P. Selinger, “Exact synthesis of multiqubit clifford+T circuits”, *Phys. Rev. A*, vol. 87, p. 032 332, 3 2013.

[155] A. N. Glaudell, N. J. Ross, J. van de Wetering, and L. Yeh, “Qutrit metaplectic gates are a subset of clifford+t”, en, Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2022.

[156] A. Zulehner and R. Wille, “Introducing design automation for quantum computing”, in *Introducing Design Automation for Quantum Computing*, Springer International Publishing, 2020, pp. 105–127.

[157] X. Gao, P. Appel, *et al.*, “On the role of entanglement in qudit-based circuit compression”, *Quantum*, vol. 7, p. 1141, 2023.

[158] A. Litteken, L. M. Seifert, *et al.*, “Qompress: Efficient compilation for ququarts exploiting partial and mixed radix operations for communication reduction”, ASPLOS 2023, pp. 646–659, 2023.

[159] A. Chiesa, F. Petiziol, *et al.*, “Embedded quantum-error correction and controlled-phase gate for molecular spin qubits”, *AIP Advances*, vol. 11, no. 2, p. 025 134, 2021.

[160] F. Petiziol, A. Chiesa, *et al.*, “Counteracting dephasing in molecular nanomagnets by optimized qudit encodings”, *npj Quantum Information*, vol. 7, no. 1, 2021.

[161] D. Janković, J.-G. Hartmann, M. Ruben, and P.-A. Hervieux, “Noisy qudit vs multiple qubits : Conditions on gate efficiency”, 2023.

[162] S. Clark, “Valence bond solid formalism for d-level one-way quantum computation”, *Journal of Physics A: Mathematical and General*, vol. 39, no. 11, pp. 2701–2721, 2006.

[163] S. Sieranoja and P. Fränti, “Adapting k-means for graph clustering”, *Knowl. Inf. Syst.*, vol. 64, no. 1, pp. 115–142, 2022.

[164] X. Wang, L. Gu, H.-W. Lee, and G. Zhang, “Quantum context-aware recommendation systems based on tensor singular value decomposition”, 2021.

[165] G. H. Low and I. L. Chuang, “Hamiltonian simulation by qubitization”, 2019.

[166] A. M. Childs, D. Maslov, *et al.*, “Toward the first quantum simulation with quantum speedup”, 2018.

[167] A. Y. Kitaev, “Quantum measurements and the abelian stabilizer problem”, *Electron. Colloquium Comput. Complex.*, 1995.

[168] A. W. Harrow, A. Hassidim, and S. Lloyd, “Quantum algorithm for linear systems of equations”, 2009.

[169] S. Lloyd, M. Mohseni, and P. Rebentrost, “Quantum principal component analysis”, 2014.

[170] P. Niemann, R. Datta, and R. Wille, “Logic synthesis for quantum state generation”, 2016.

[171] F. Mozafari, G. D. Micheli, and Y. Yang, “Efficient deterministic preparation of quantum states using decision diagrams”, 2022.

[172] F. Mozafari, Y. Yang, and G. D. Micheli, “Efficient preparation of cyclic quantum states”, 2022.

[173] X.-M. Zhang, T. Li, and X. Yuan, “Quantum state preparation with optimal circuit depth: Implementations and applications”, 2022.

[174] M. Krenn, M. Malik, *et al.*, “Automated search for new quantum experiments”, 2016.

[175] S. Hillmich, A. Zulehner, *et al.*, “Approximating decision diagrams for quantum circuit simulation”, *ACM Transactions on Quantum Computing*, 2022.

[176] A. Cabello, “Bell’s theorem with and without inequalities for the three-qubit Greenberger-Horne-Zeilinger and W states”, 2002.

[177] Y.-M. Di and H.-R. Wei, “Synthesis of multivalued quantum logic circuits by elementary gates”, 2013.

[178] W. Zi, Q. Li, and X. Sun, *Optimal synthesis of multi-controlled qudit gates*, 2023. arXiv: 2303.12979 [quant-ph].

[179] S. Bullock, D. O’Leary, and G. Brennen, “Asymptotically optimal quantum circuits for d-level systems”, *Phys. Rev. Lett.*, vol. 94, no. 23, 2005.

[180] A. Botea, A. Kishimoto, and R. Marinescu, “On the complexity of quantum circuit compilation”, in *Symp. on Combinatorial Search*, 2018.

[181] S. Bullock and I. Markov, “An arbitrary two-qubit computation in 23 elementary gates or less”, 2003, pp. 324–329.

[182] K. N. Smith and M. A. Thornton, “Entanglement in higher-radix quantum systems”, IEEE Computer Society, 2019, pp. 114–119.

[183] L. E. Heyfron and E. T. Campbell, “A quantum compiler for qudits of prime dimension greater than 3.”, *arXiv: 1902.05634*, 2019.

[184] A. Glaudell, “Quantum compiling methods for fault-tolerant gate sets of dimension greater than two”, Ph.D. dissertation, University of Maryland, College Park, 2019.

[185] E. A. Martinez, T. Monz, *et al.*, “Compiling quantum algorithms for architectures with multi-qubit gates”, *New Journal of Physics*, vol. 18, no. 6, p. 063 029, 2016.

[186] N. Friis, V. Dunjko, W. Dür, and H. J. Briegel, “Implementing quantum control for unknown subroutines”, *Phys. Rev. A*, vol. 89, no. 3, 2014.

[187] K. Bharti, A. Cervera-Lierta, *et al.*, “Noisy intermediate-scale quantum algorithms”, *Rev. Mod. Phys.*, vol. 94, p. 015 004, 1 2022.

[188] M. Cerezo, A. Arrasmith, *et al.*, “Variational quantum algorithms”, *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[189] S. Sim, P. D. Johnson, and A. Aspuru-Guzik, “Expressibility and entangling capability of parameterized quantum circuits for hybrid quantum-classical algorithms”, *Advanced Quantum Technologies*, vol. 2, no. 12, p. 1 900 070, 2019.

[190] C. Spengler, M. Huber, and B. C. Hiesmayr, “Composite parameterization and haar measure for all unitary and special unitary groups”, *Journal of Mathematical Physics*, vol. 53, no. 1, p. 013 501, 2012.

[191] B. C. Sawyer and K. R. Brown, “Wavelength-insensitive, multispecies entangling gate for group-2 atomic ions”, 2021.

[192] X. Wang, C.-s. Yu, and X. X. Yi, “An alternative quantum fidelity for mixed states of qudits”, *Physics Letters A*, vol. 373, pp. 58–60, 2008.

[193] D. C. Liu and J. Nocedal, “On the limited memory BFGS method for large scale optimization”, *Mathematical Programming*, vol. 45, no. 1-3, pp. 503–528, 1989.

[194] P. Virtanen, R. Gommers, *et al.*, “SciPy 1.0: Fundamental algorithms for scientific computing in python”, *Nature Methods*, vol. 17, pp. 261–272, 2020.

[195] C. R. Harris *et al.*, “Array programming with NumPy”, *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[196] M. Larocca, N. Ju, *et al.*, “Theory of overparametrization in quantum neural networks”, 2021. arXiv: 2109.11676.

[197] R. D. da Silva, E. Pius, and E. Kashefi, “Global quantum circuit optimization”, *arXiv preprint arXiv:1301.0351*, 2013.

[198] R. Duncan, A. Kissinger, S. Perdrix, and J. van de Wetering, “Graph-theoretic simplification of quantum circuits with the ZX-calculus”, *Quantum*, vol. 4, p. 279, 2020.

[199] A. Kissinger and J. van de Wetering, “Reducing the number of non-clifford gates in quantum circuits”, *Physical Review A*, vol. 102, no. 2, 2020.

[200] Q. Wang, *Qufinite zx-calculus: A unified framework of qudit zx-calculi*, 2021.

[201] K. Mølmer and A. Sørensen, “Multiparticle entanglement of hot trapped ions”, *Phys. Rev. Lett.*, vol. 82, pp. 1835–1838, 1999.

[202] G. Brennen, D. O’Leary, and S. Bullock, “Criteria for exact qudit universality”, *Phys. Rev. A*, vol. 71, no. 5, 2005.

[203] D. P. O’Leary, G. K. Brennen, and S. S. Bullock, “Parallelism for quantum computation with qudits”, *Phys. Rev. A*, vol. 74, no. 3, 2006.

[204] E. O. Kiktenko, A. S. Nikolaeva, *et al.*, “Scalable quantum computing with qudits on a graph”, *Phys. Rev. A*, vol. 101, p. 022 304, 2 2020.

[205] P. J. Low, B. M. White, *et al.*, “Practical trapped-ion protocols for universal qudit-based quantum computing”, *Phys. Rev. Res.*, vol. 2, no. 3, 2020.

[206] D. M. Nicol, “Expected performance of m-solution backtracking”, *SIAM J. on Comp.*, vol. 17, no. 1, pp. 114–127, 1988.

[207] M. Huber and J. I. de Vicente, “Structure of multidimensional entanglement in multipartite systems”, *Phys. Rev. Lett.*, vol. 110, no. 3, 2013.

[208] Microsoft, *Q# Language Specification*, 2020.

[209] B. Bichsel, M. Baader, T. Gehr, and M. Vechev, “Silq: A high-level quantum language with safe uncomputation and intuitive semantics”, New York, NY, USA: Association for Computing Machinery, 2020.

[210] A. S. Green, P. L. Lumsdaine, *et al.*, “An introduction to quantum programming in quipper”, in *Reversible Computation*. Springer Berlin Heidelberg, 2013, pp. 110–124.

[211] A. J. Abhari, A. I. Faruque, *et al.*, “Scaffold: Quantum programming language”, 2012.

[212] A. W. Cross, L. S. Bishop, J. A. Smolin, and J. M. Gambetta, *Open quantum assembly language*, 2017. arXiv: 1707.03429 [quant-ph].

[213] R. S. Smith, M. J. Curtis, and W. J. Zeng, *A practical quantum instruction set architecture*, 2017. arXiv: 1608.03355 [quant-ph].

[214] W. van Dam, M. Mykhailova, and M. Soeken, *Using azure quantum resource estimator for assessing performance of fault tolerant quantum computation*, 2024. arXiv: 2311.05801 [quant-ph].

[215] M. Stęchły and PsiQuantum, *Bartiq*, 2024.

[216] N. Killoran, J. Izaac, *et al.*, “Strawberry fields: A software platform for photonic quantum computing”, *Quantum*, vol. 3, p. 129, 2019.

[217] N. Khammassi, G. G. Guerreschi, *et al.*, *Cqasm v1.0: Towards a common quantum assembly language*, 2018. arXiv: 1805.09607 [quant-ph].

[218] C. Roberts, A. Milsted, *et al.*, *Tensornetwork: A library for physics and machine learning*, 2019. arXiv: 1905.01330 [physics.comp-ph].

List of Figures

List of Tables