university of groningen

faculty of science and engineering

Effectiveness of quantum annealing and the quantum approximate optimization algorithm for solving audio quantization problems

Internship MSc Applied Mathematics

Student: M. Repášová
Company/Institute: TNO
Internal UoG supervisor/first assessor: Dr. Ir. H. J. van Waarde
Internal UoG second assessor: Prof. K. M. Camlibel
External supervisors: Dr. J. Verbree and Dr. N.M.P. Neumann

Abstract

Quantum computing is an emerging field of computer science and engineering. It has the potential to solve problems beyond the ability of classical computers using the unique qualities of quantum mechanics. We are in the Noisy Intermediate Scale Quantum (NISQ) era of quantum computing, where only small-scale and error-prone quantum hardware is available. Nonetheless, a wide range of algorithms and heuristics have been proposed to take advantage of the current era of quantum hardware. In this project, we study the efficacy of two such approaches, Quantum Annealing (QA) and the Quantum Approximate Optimization Algorithm (QAOA), on an example of audio quantization. This use case was first studied in [1] where the authors have found that QA, using D-Wave's 2000Q annealer, finds higher quality solutions than a comparable classical algorithm- simulated annealing. We expand on their work and compare the performance of simulated annealing, QA using D-Wave's newer annealer Advantage, and the QAOA. Contrary to their finding, we find that simulated annealing finds higher quality solutions than QA. Moreover, we find that QAOA performs the worst.

Contents

Notation

In this section, we introduce the notation used in this report.

We denote by $\mathbb{N}$ the set of natural numbers, by $\mathbb{Z}$ the set of integers, by $\mathbb{Z}_+$ the set of nonnegative integers, by $\mathbb{R}$ the set of real numbers, and by $\mathbb{C}$ the set of complex numbers. Moreover, we denote by $\{0,1\}^n$ the set of vectors in $\mathbb{R}^n$ with elements either 1 or 0. Given a set $\mathbb{U} \subset \mathbb{R}$, we define the set $\mathbb{U}^N \subset \mathbb{R}^N$ as

$$\mathbb{U}^N := \underbrace{\mathbb{U} \times \cdots \times \mathbb{U}}_{N \text{ times}}.$$

For $u\colon \mathbb{Z}_+ \to \mathbb{R}^m$ and $T \in \mathbb{Z}_+$, we define the set $\{u(t)\}_{t=0}^T := \{u(0), u(1), \ldots, u(T)\}$.

Moreover, for $x, y \in \{0,1\}$ we denote by $x \oplus y$ addition modulo 2. Given sequences (x_i) and (y_i) we define the sum notation

$$\sum_{i \neq j}^{n} x_i y_j := \sum_{i=1}^{n} \sum_{j \in \{1,2,\ldots,n\} \setminus \{i\}} x_i y_j.$$

Consider the matrices $A \in \mathbb{C}^{n \times m}$ and $B \in \mathbb{C}^{l \times k}$. Then, the Kronecker product $A \otimes B \in \mathbb{C}^{nl \times mk}$ is given by the partitioned matrix

$$A \otimes B = \begin{bmatrix} A_{11}B & A_{12}B & \ldots & A_{1m}B \\ \vdots & \vdots & \ddots & \vdots \\ A_{n1}B & A_{n2}B & \ldots & A_{nm}B \end{bmatrix}.$$

Here A_{ij} denotes the element of A in the i-th row and j-th column. Given n matrices $M_1, M_2, \ldots, M_n$, we denote by $\bigotimes_{i=1}^n M_i$ the Kronecker product $M_1 \otimes M_2 \otimes \cdots \otimes M_n$.

Moreover, given a matrix $M \in \mathbb{C}^{N \times N}$ we denote by $M^\dagger$ its Hermitian transpose and by

$$M^{\otimes n} = \underbrace{M \otimes M \otimes \cdots \otimes M}_{n \text{ times}}$$

the n-times Kronecker product of M with itself.

We will use the Dirac vector notation "$|\cdot\rangle$" for quantum states. For a state $|\psi\rangle$, we denote by $\langle\psi|$ its covector, that is, its Hermitian transpose. Given states $|\psi\rangle$ and $|\phi\rangle$, we denote by $\langle\psi|\phi\rangle$ the standard Hermitian product of these vectors. Lastly, for an operator A, we denote by $\langle\psi|A|\phi\rangle$ the standard Hermitian inner product of $|\psi\rangle$ and $A|\phi\rangle$, or equivalently, the inner product of $A^\dagger|\psi\rangle$ and $|\phi\rangle$.

Lastly, we define big-O notation as in [2]. Consider $f, g\colon \mathbb{N} \to \mathbb{N}$. We say that $f(n) = \mathcal{O}(g(n))$ if and only there exist $\alpha, n_o \in \mathbb{N}$ such that

$$\forall n \geq n_o \implies f(n) \leq \alpha g(n).$$

Then, $g(n)$ is an asymptotic upper bound for $f(n)$, and we say that f is of the order of g. Informally, $f(n) = \mathcal{O}(g(n))$ means that f grows as g or slower.

1 Introduction

1.1 TNO

TNO, the Dutch organization for applied scientific research (Nederlandse Organisatie voor Toegepast Natuurwetenschappelijk Onderzoek), is an independent non-profit research organization. Founded in 1932, TNO focuses on applied research that supports innovation, economic competitiveness, and societal well-being in the Netherlands and beyond. TNO works across diverse domains, including sustainability, health, safety, and digital technologies.

This project was done in TNO's ACQuA (Applied Cryptography & Quantum Algorithms) department. ACQuA consists of an expert group of mathematicians, computer scientists, and physicists that creates state-of-the-art software, algorithms, and strategies for today and the quantum age of the near future.

1.2 Project description

The modern theory of quantum mechanics was developed in the 1920s to explain perplexing physical phenomena observed at atomic scales. In 1980, Paul Benioff introduced the quantum Turing machine [3], which uses quantum theory to describe a simplified computer. Independently, Manin in 1980 [4] and Feymann in 1982 [5] came up with the idea that a quantum computer might simulate quantum systems more efficiently than any classical computer could [6]. Analogous to the way a classical computer is built from an electronic circuit containing wires and logic gates, a quantum computer can be built from a quantum circuit containing wires and elementary quantum gates to carry around and manipulate quantum information.

Interest in quantum computing stems from the idea of a *quantum speedup*. It is based on the idea that a quantum algorithm could exploit properties of qubits like entanglement and superposition to carry out computations faster than any classical algorithm could. In 1994, Shor in [7] developed a quantum algorithm capable of factoring large integers exponentially faster than any known classical algorithm. Another famous quantum algorithm is Grover's search algorithm, introduced in 1996 in [8], that offers a quadratic speedup over classical search algorithms.

However, these algorithms require a fully error-corrected quantum device capable of handling in the order of 10^5 qubits [9] that are yet to be developed. One of the biggest obstacles in developing large-scale quantum devices is decoherence, a physical phenomenon occurring when qubits interact with their surrounding environment and lose their coherent features [10].

Currently, we are in a Noisy Intermediate Scale Quantum (NISQ) era where only small-scale and error-prone quantum hardware is available. Despite these limitations, a wide

range of algorithms have been proposed to take advantage of the current era of quantum computing. Among them, variational quantum algorithms are a promising class of hybrid classical-quantum algorithms that combine classical algorithms with small-scale quantum hardware. One of the popular variational quantum algorithms is the Quantum Approximate Optimization Algorithm (QAOA), designed to solve combinatorial optimization problems by iteratively refining parameters of a parametrized quantum circuit using a classical optimizer. Unlike variational quantum algorithms that use gate-based parametrized quantum circuits, Quantum Annealing (QA) is a different paradigm of quantum computing compatible with NISQ devices. QA operates by gradually evolving a state via a quantum system defined by a time-dependent Hamiltonian that guides it to a low-energy configuration that corresponds to a (near-)optimal solution.

Both QA and QAOA are designed to solve combinatorial optimization problems that consist of finding an optimal solution from a finite set of objects. These problems are NP-hard; that is, there is no known efficient algorithm that can solve them. There have been many theoretical and experimental studies on these methods. However, evidence of a non-problem-specific quantum speedup using them is yet to be found.

In [1], the authors studied the efficacy of QA to solve combinatorial optimization problems stemming from model predictive control for finite input systems. In both cases they have studied, QA outperformed the classical algorithm- simulated annealing. One of these cases was the problem of quantizing audio signals.

Recording an audio signal consists of two steps. First, we sample an analog signal with a certain sampling rate. Then, in order to store it digitally, this discrete sampled signal is quantized. The process of quantization consists of converting a finely sampled digital signal into a finite set of discrete values. These discrete values are often bitvectors, that is, vectors with elements either 0 or 1. The accuracy of quantization depends on the length of these vectors, which is determined by the number of bits we use. For n bits, we have 2^n different bitvectors to approximate the audio signal with. Typical values of bits used for quantization are $8, 16, 24$, or even 32 for fine resolution. For example, CDs use 16-bit depth, whereas DVDs and Blu-Ray discs use 32-bit depth.

In this project, we will also study the efficacy of quantum computing for the problem of audio quantization. We will expand on the work of [1] and add to the comparison the QAOA.

1.3 Research Questions

Our first research question is to verify results from [1] for their use case of audio quantization and add QAOA to the comparison.

Our second research question is to compare the performance of QA, QAOA, and SA with respect to the scaling of the problem size on an example of audio quantization.

1.4 Outline

In Chapter 2, we introduce combinatorial optimization problems and the three approaches to solving them that we consider in this project. That is, simulated annealing, QA, and the QAOA. To put our results in a context, we also provide a brief overview of recent results on the analysis and performance of QA and the QAOA.

Then, in Chapter 3, we introduce the problem of audio quantization using perception filters. We show that it is an example of model predictive control for finite input systems. Moreover, following the work of [1], we show how these problems can be transformed into a special class of combinatorial optimization, Quadratic Unconstrained Binary Optimization (QUBO).

In Chapter 4, we discuss our numerical results. First, we tune the parameters of each of the three methods we consider. Then, we compare their performance with respect to the scale of the problem size. Moreover, we test them on the example from [1].

Lastly, in Chapter 6, we conclude this report.

2 Background theory

In this chapter, we discuss combinatorial optimization problems and approaches to solving them. Namely, we consider a classical algorithm- simulated annealing, a quantum heuristic- Quantum Annealing, and a hybrid classical-quantum algorithm- Quantum Approximate Optimization Algorithm. For background theory on quantum computing, we refer the reader to Appendix A.

2.1 Combinatorial optimization problems

Combinatorial optimization is a class of computational problems that consists of optimizing a function within a set of finite possible values. It has a diverse set of applications across countless fields such as logistics, supply chain optimization, bio-informatics, game theory, and machine learning, and is actively researched in Operations Research, Mathematics, and Computer Science [11].

In this section, we introduce combinatorial optimization and a type of combinatorial optimization commonly researched, quadratic unconstrained binary optimization.

We define a *combinatorial optimization* problem as an optimization problem of the form

$$\min_{x \in S} f(x) \tag{1}$$

where f is some cost function we aim to minimize over the set S. Here S is a finite set such that $S = \{s_1, s_2, \ldots, s_N\}$ for some $N \in \mathbb{N}$. Commonly, this set is expressed using binary variables, that is, $S = \{0,1\}^n$ for some $n \in \mathbb{N}$. We denote by F^* the set of optimal solutions and by $F_{\min} = f(x)$ for $x \in F^*$ the optimal cost.

Moreover, we define a neighborhood function $\mathcal{N}\colon S \to 2^S$ as a function that assigns to each $i \in S$ a set $S_i \subseteq S$ of solutions that is in some sense close to i. How "close to" is defined depends on the user and the particular application of interest.

Due to manufacturing constraints, implementations of QA and the QAOA focus on a specific type of combinatorial optimization, called the Quadratic Unconstrained Binary Optimization.

We have that *Quadratic Unconstrained Binary Optimization* (QUBO) is a type of combinatorial optimization where the cost function $f(x)$ is a quadratic function of binary variables. We define it as the following optimization problem

$$\min_{x \in \{0,1\}^n} x^\top Q x, \tag{2}$$

where $Q \in \mathbb{R}^{n \times n}$. Note that this form includes all quadratic functions of binary variables. To show this, consider some quadratic function of binary variables of the form $x^\top A x + bx + c$ where $A \in \mathbb{R}^{n \times n}$, $b \in \mathbb{R}^n$ and $c \in \mathbb{R}$. First of all, the constant term is

independent of the variable to minimize over and thus can be neglected in solving for the minimizer. Moreover, (2) includes linear terms as well. This is because for $x = \begin{bmatrix} x_1 & x_2 & \dots & x_n \end{bmatrix}^\top \in \{0,1\}^n$ we have that $x_i^2 = x_i$ and hence

$$\min_{x\in\{0,1\}^n} x^\top Ax + bx = \min_{x\in\{0,1\}^n} \sum_{i,j=1}^{n} A_{ij}x_i x_j + \sum_{i=1}^{n} b_i x_i = \min_{x\in\{0,1\}^n} x^\top Qx$$

for $Q \in \mathbb{R}^{n\times n}$ such that $Q_{ij} = A_{ij}$ for $i \neq j$ and $Q_{ii} = A_{ii} + b_i$ for all $i, j \in \{1, 2, \dots, n\}$.

2.2 Simulated Annealing

In this section, we introduce the Simulated Annealing algorithm (SA) using [12] and [13]. Simulated annealing is a heuristic algorithm. That is, it is an algorithm that has weak or nonexistent theoretical guarantees on runtime and/or solution quality. It is one of the best-known local search algorithms (sometimes referred to as Monte Carlo algorithms) and is widely used, since it is a practical and flexible method.

Simulated annealing was first introduced in the 1980s by Kirkpatrick et al. [14] and independently by Černý [15]. It was heavily inspired by an analogy between the physical annealing process of solids and the problem of solving large combinatorial problems.

In condensed matter physics, annealing is known as a thermal process for obtaining low-energy states of a solid in a heat bath. It consists of two steps. First, the temperature of the heat bath is increased to a maximum value at which the solid melts. Then, the temperature is decreased slowly until the particles arrange themselves in the ground state of the solid [12], that is, a state with the lowest internal energy.

Metropolis et al. in [16] introduced a simple algorithm for simulating the process. It consists of a so-called Metropolis criterion for acceptance of a new state. Namely, given a state i with energy E_i, generate a subsequent state j with energy E_j by applying a perturbation mechanism to the state i. Then, if $E_j - E_i \leq 0$, accept as the new state j. Otherwise, accept j as the new state with the probability $\exp\left(\frac{E_j - E_i}{k_B T}\right)$. Here k_B is a constant, called the Boltzmann constant, and T denotes the temperature of the heat bath.

Simulated annealing uses this criterion for acceptance of a new state with decreasing values of a control parameter c_k, which serves as the temperature. One iteration of the simulated annealing algorithm consists of the following 3 steps.

(1) Given initial state i, generate a new state $j \in S_i$ where S_i is some user-defined neighborhood of i.

(2) If $f(j) \leq f(i)$ replace i with j. Otherwise replace i with j with the probability $\exp\left(\frac{f(i) - f(j)}{c_k}\right)$.

(3) Repeat steps (1) and (2) for $k = 1, 2, \ldots, n_s$ where n_s is the number of sweeps.

We repeat this process until a certain stopping criterion is reached. This can be $c_k < \epsilon$ for some $\epsilon > 0$, or we iterate a fixed number of times. Here, c_k is a user-defined control parameter that decreases as k increases. Common choice is

$$c_{k+1} = \alpha c_k$$

where $\alpha \in [0.8, 1)$ [12]. Given its probabilistic nature, it is common to repeat this iterative process multiple times and pick the best solutions out of many. We call the parameters determining how many times we repeat this process the number of reads n_r.

2.3 Quantum Annealing

Quantum annealing is a heuristic search approach to solving problems in combinatorial optimization. Similar to simulated annealing being based on a physical process of thermal annealing, quantum annealing is based on the physical process of quantum annealing.

In Figure 1, taken from [17], we illustrate two types of movements from a local minimum of a computationally hard optimization problem.

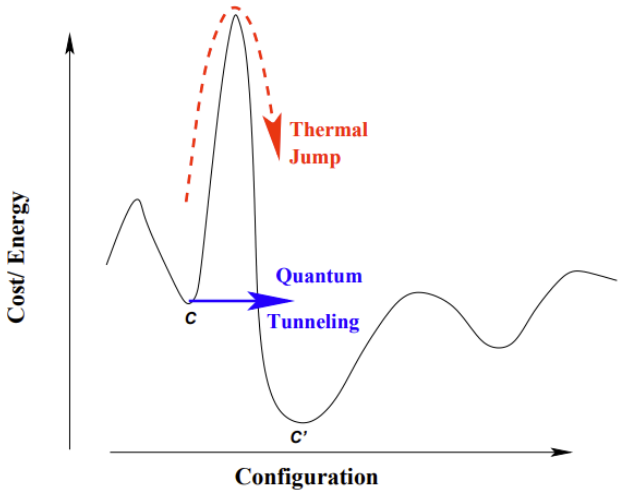


Figure 1: Thermal jump vs. Quantum Tunneling

Consider a cost function of a computationally hard problem with multiple local minima. Then, in Figure (1), C corresponds to a shallower local minimum and C' to a deeper minimum of the cost function. Simulated annealing uses thermal jump-like fluctuations to attempt to move from a suboptimal minimum, whereas quantum annealing can also use tunneling-like fluctuations.

In this section, we first introduce Adiabatic Quantum Computing (AQC) and then explain the Quantum Annealing (QA) method as it is commonly used in practice.

2.3.1 Adiabatic Quantum Computing

Adiabatic quantum computing (AQC) was introduced by Farhi et. al in [18]. Unlike the gate-based model of quantum computing with discrete steps, AQC models qubits gradually evolving according to certain forces represented by Hamiltonians. The difference between gate-based quantum computing and AQC is illustrated in Figure 2 taken from [19].

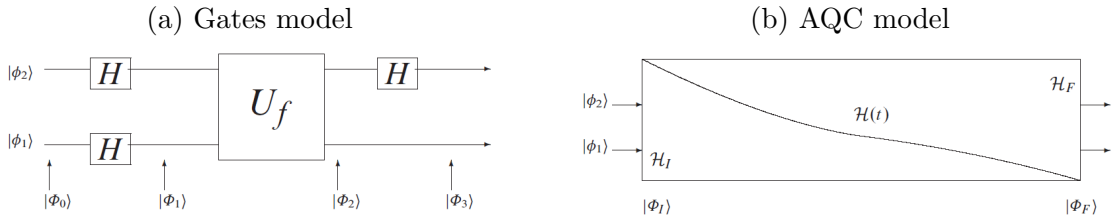


Figure 2: Two types of quantum computing

We have that a *Hamiltonian* is an operator that corresponds to the total energy of a quantum system described by a Hermitian matrix $H = H^\dagger$.

Consider a dynamical quantum system described by the time-dependent Hamiltonian $H_t \in \mathbb{C}^{N\times N}$. We have the following definitions [19].

Any observable state s has an associated *energy* which is a real scalar vector dependent on H_t. The eigenstates of H_t correspond to the observable states of the system. If $|\phi_t\rangle$ is an eigenstate with energy λ then $H_t |\phi_t\rangle = \lambda |\phi_t\rangle$. The *energy spectrum* is the set of all possible energies in the system, of size at most N. Two or more observable states with the same energy are said to be *degenerate*. The *ground state* s_g is an observable state having minimal energy over all basis states. A state that is not a ground state is said to be *excited*. The *first excited state* is the eigenstate with the lowest energy among all excited states.

In short, an AQC algorithm to solve an optimization problem P with objective function $f(x)$ works on a register Q of n qubits having superposition state $|\phi_t\rangle$ at time t. The algorithm is described by a time-varying Hamiltonian $H(t)$ such that

$$H(t) = s(t)H_I + (1 - s(t))H_F, \tag{3}$$

where H_I is an initial Hamiltonian chosen such that its ground state is easy to find. H_F is the final (or problem) Hamiltonian chosen such that the ground state of H_F corresponds to the optimal solution of P. Lastly, $s(t)$ is an adiabatic evolution path function that descreases from 1 to 0 over $t \in [0, t_f]$.

A general AQC method proceeds as follows. First, we prepare an initial state such that it is the ground state of H_I. Then, over $[0, t_f]$ we evolve the state according to the

Schrödinger equation

$$i\frac{d}{dt}\left|\phi(t)\right\rangle = H(t)\left|\phi(t)\right\rangle.$$

with the Hamiltonian $H(t)$ (3) to obtain the state $|\phi_F\rangle$. As a last step of the method, we measure the resulting state $|\phi_F\rangle$. By the Adiabatic theorem, the state $|\phi_F\rangle$ will correspond to the ground state of H_F with a high probability.

2.3.2 Adiabatic theorem

Why AQC works is explained by the Adiabatic theorem. In this section, we introduce this theorem.

In short, it states that if we start with a ground state of the initial Hamiltonian and evolve it slowly to the final Hamiltonian, the state will remain a ground state. Consider a time-independent Hamiltonian of a quantum system H. Then, the time evolution of the quantum system is given by the Schrödinger equation

$$i\frac{d}{dt}\left|\psi(t)\right\rangle = H\left|\psi(t)\right\rangle.$$

We have that given the initial state $|\psi(0)\rangle$, the solution to the equation is

$$\left|\psi(t)\right\rangle = e^{-iHt}\left|\psi(0)\right\rangle.$$

From this we can immediately see that given eigenvalue λ with eigenstate $|\psi_\lambda\rangle$ such that $H\left|\psi_\lambda\right\rangle = \lambda\left|\psi_\lambda\right\rangle$, we get for $|\psi(0)\rangle = |\psi_\lambda\rangle$ that $|\psi(t)\rangle = e^{-Ht}\left|\psi_\lambda\right\rangle = e^{-i\lambda t}\left|\psi_\lambda\right\rangle$. Hence, if the system begins in an eigenstate, it remains in the eigenstate and only acquires a phase $e^{-i\lambda t}$.

If the Hamiltonian of a quantum system varies in time, the time evolution of the quantum system becomes more complicated. The *Adiabatic theorem* states that if the change in the Hamiltonian occurs sufficiently slowly, the dynamics remain relatively simple, and if the system begins in an eigenstate, it remains close to the eigenstate. More specifically, in [20], the theorem is described as follows. Let $|\psi_{\lambda_k}(t)\rangle$ be the eigenstate of $H(t)$ such that

$$H(t)\left|\psi_{\lambda_k}(t)\right\rangle = \lambda_k(t)\left|\psi_{\lambda_k}(t)\right\rangle,$$

where $\lambda_k(t)$ are the corresponding eigenvalues and k labels the eigenstate (such that $k = 0$ corresponds to the ground state). We define the minimum gap between two eigenvalues as

$$g_{\min} := \min_{0\leq t\leq T}\left[\lambda_1(t) - \lambda_0(t)\right]$$

and the matrix element of $\frac{dH}{dt}$ between the two corresponding eigenstates as

$$\left\langle \frac{dH}{dt} \right\rangle_{1,0} = \left\langle \psi_{\lambda_1}(t) \middle| \frac{dH}{dt} \middle| \psi_{\lambda_0}(t) \right\rangle .$$

The Adiabatic Theorem states that if we prepare the system at time $t = 0$ in its ground state $|\psi_{\lambda_0}\rangle$ and let it evolve under the Hamiltonian $H(t)$, then

$$|\langle \psi_{\lambda_0}(T) | \psi(T) \rangle|^2 \geq 1 - \epsilon^2$$

provided that

$$\frac{\left| \left\langle \frac{dH}{dt} \right\rangle_{1,0} \right|}{g_{\min}^2} \leq \epsilon$$

where $\epsilon \ll 1$. A precise mathematical statement of the Adiabatic theorem and adiabatic theory behind it is beyond the scope of this project. For a detailed treatment of quantum adiabatic theory, we refer the reader to, for example, [21].

2.3.3 Quantum annealing

In this section, we introduce quantum annealing (QA) and discuss how it is used in practice using quantum annealer hardware from D-Wave. Quantum annealing can be viewed as a relaxation of the AQC model, where the conditions of adiabacity are not necessarily met, resulting in a heuristic variational quantum algorithm [22].

Consider the combinatorial optimization problem (1). Just as in AQC, QA is based on evolving a quantum system using a time-dependent Hamiltonian $H(t)$.

The time-varying Hamiltonian for QA is

$$H(t) = A(\tau)H_I + B(\tau)H_P$$

where $\tau = \frac{t}{t_a}$ for $0 \leq t \leq t_a$ and t_a is the total annealing time. The path functions $A(s)$, $B(s)$ control the transition from H_I to H_P over the time interval $[0, t_a]$. In current D-Wave's QPUs, these functions are related by $B(s) = 1 - A(s)$. The problem Hamiltonian H_P is chosen such that its ground state encodes the minimum of the cost function $f(x)$ from (1). Whereas the initial Hamiltonian H_I in QA is referred to as the disordering Hamiltonian that introduces kinetic energy to the annealing process in the form of quantum fluctuations of the solution space [19]. We often choose it such that its ground state is easy to compute.

QA consists of 4 stages. First, the quantum system is initialized according to $H(t)$, and qubits are placed in an initial superposition state that corresponds to the ground state of H_I. Then, we anneal by evolving the system over time t_a. At the end of the annealing process, qubit values are read, yielding a solution to the input. Lastly, we resample. Since measurements of solutions of a quantum system are probabilistic, the computation

does not always finish in a ground state. Given the relatively high initialization time, it is cost-effective to repeat the anneal-readout cycle many times per input [2]. We denote by n_r the number of reads, that is, the number of times we repeat this process.

Due to manufacturing constraints, current experimental devices can only handle 2-local interactions, i.e., QUBO problems [23].

Now we describe how one sets up the problem Hamiltonian H_P for a given QUBO problem (2). We aim to find a Hamiltonian H_F such that

$$H_F \left|\psi_x\right\rangle = f(x) \left|\psi_x\right\rangle$$

that is, each eigenvector $\left|\psi_x\right\rangle$ is matching a classical solution $x \in \{0,1\}^n$ with corresponding eigenvalue $f(x)$. Then, the ground state of the Hamiltonian corresponds to the minimum of $f(x)$.

We have that for the QUBO cost function $f(x) = \sum_{i,j} x_i Q_{i,j} x_j$, the Hamiltonian operator is

$$H_F = \frac{1}{4}\sum_{i\neq j}^{n} Q_{ij}\sigma_i^z\sigma_j^z - \frac{1}{2}\sum_{i}^{n}\left(Q_{ii} + \sum_{j}^{n} Q_{ij}\right)\sigma_i^z + \frac{1}{2}\sum_{i}^{n} Q_{ii} + \frac{1}{4}\sum_{i\neq j}^{n} Q_{ij}, \tag{4}$$

where by σ_i^z we denote the Pauli-Z gate $\sigma^z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}$ acting on a qubit i. That is,

$$\sigma_i^z = I^{\otimes(i-1)} \otimes \sigma^z \otimes I^{\otimes(n-i)}.$$

We derive (4) as follows. We have that

$$\sigma_i^z \left|\psi_x\right\rangle = (-1)^{x_i} \left|x\right\rangle = (1 - 2x_i) \left|x\right\rangle$$

and hence

$$x_i \left|\psi_x\right\rangle = \frac{I - \sigma_i^z}{2} \left|\psi_x\right\rangle .$$

Then,

$$\begin{aligned}
f(x)\left|\psi_x\right\rangle &= \left(\sum_{i}^{n} Q_{ii}x_i + \sum_{i\neq j}^{n} Q_{ij}x_i x_j\right)\left|\psi_x\right\rangle \\
&= \left(\sum_{i}^{n} Q_{ii}\frac{I-\sigma_i^z}{2} + \sum_{i\neq j}^{n} Q_{ij}\frac{I-\sigma_i^z}{2}\frac{I-\sigma_j^z}{2}\right)\left|\psi_x\right\rangle \\
&= \left(\frac{1}{4}\sum_{i\neq j}^{n} Q_{ij}\sigma_i^z\sigma_j^z - \frac{1}{2}\sum_{i}^{n}\left(Q_{ii} + \sum_{j}^{n} Q_{ij}\right)\sigma_i^z + \frac{1}{2}\sum_{i}^{n} Q_{ii} + \frac{1}{4}\sum_{i\neq j}^{n} Q_{ij}\right)\left|\psi_x\right\rangle \\
&= H_F \left|\psi_x\right\rangle .
\end{aligned}$$

Note, for optimization purposes, the constant terms $\frac{1}{2}\sum_i^n Q_{ii} + \frac{1}{4}\sum_{i\neq j}^n Q_{ij}$ are irrelevant. Hence often, for QUBO problems, we set

$$H_P = \frac{1}{4}\sum_{i\neq j}^{n} Q_{ij}\sigma_i^z\sigma_j^z - \frac{1}{2}\sum_{i}^{n}\left(Q_{ii} + \sum_{j}^{n} Q_{ij}\right)\sigma_i^z.$$

Lastly, in the implementation of quantum annealing, the initial Hamiltonian is often chosen such that

$$H_I = \sum_{i=1}^{n} \sigma_i^x,$$

where σ_i^x denotes the Pauli-X gate $\sigma^x = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ applied to the i-th qubit, that is,

$$\sigma_i^x = I^{\otimes(i-1)} \otimes \sigma^x \otimes I^{\otimes(n-i)}.$$

Then, the ground state of this Hamiltonian is $|+\rangle^{\otimes n} = \sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{2^n}} |x\rangle$, which we can prepare using the Hadamard gate.

2.3.4 Quantum annealing implementations and their performance

In this section, we discuss current trends and findings about various implementations of QA and their performance. To do so, we summarize the main findings of [24] and the review articles [23], [25], and [26].

The paradigm of quantum annealing was introduced to efficiently solve large-scale combinatorial optimization problems. QA has gained tremendous momentum since programmable quantum annealing machines, dubbed as quantum annealers, with more than a thousand qubits have been realized and commercialized [25]. However, it is still unclear whether algorithms based on quantum annealing can lead to a quantum speedup for a general class of problems.

One example where QA is shown to provide a quantum speedup is the unstructured search problem. It was shown in [20] that QA, with a particular adiabatic annealing schedule, can obtain the same quadratic speedup as Grover's search algorithm. Moreover, the authors in [24] have generalized this result. They have shown that, for a class of spin Hamiltonians whose minimal spectral gap (sometimes referred to as energy gap) is tractable to compute, one can construct an annealing schedule for adiabatic QA that achieves quadratic speedup over classical algorithms. Moreover, they have shown that this quadratic speedup is optimal for the problem class.

However, the authors in [24] conclude that, for most practical applications, the minimal spectral gap is not tractable to compute, making a quadratic speedup with adiabatic QA infeasible. Using this, they argue that adiabatic QA is unlikely to lead to a speedup. Nowadays, the research into QA is focused on non-adiabatic QA.

In [26], the authors provide a systematic review of research development trends in the field of quantum annealing and analyze how it has been implemented in different problem domains. These problem domains were grouped into 13, namely, machine learning, graphics, mathematics, routing, scheduling, computational chemistry, computational biology, security, portfolio optimization, big data, hydrology, database, and sensors. They categorize quantum annealing into four types: basic, hybrid, reverse, and improved QA. The hybrid QA is based on the combination of basic QA and classical methods to handle optimization problems. Meanwhile, the reverse QA is based on the concept that if the system is initialized in the state according to the local minimum of the objective function, then the interaction of quantum and thermal fluctuations can help the state escape the energy trap during reverse annealing. In this method, the quantum fluctuation first increases and only then decreases. Moreover, the improved QA adds certain parameters to improve the quality of a previously conducted QA [26].

In [22], the authors conclude that, based on the literature they have reviewed, the empirical evidence of QA performance is highly dependent on the problem being addressed. They find that, regardless of the application field, only relatively small problem sizes can currently be addressed efficiently on actual hardware. Additionally, in their review article [23], the authors conclude that to the best of their knowledge, there is no industrial application where QA outperforms classical heuristics. Further improvements to QA hardware will be necessary for QA to be applicable in practice.

2.4 QAOA

The Quadratic Approximate Optimization Algorithm (QAOA) is a hybrid gate-based quantum algorithm that was first introduced in 2014 by E. Farhi, J. Goldstone, and S. Gutmann to solve the MAXCUT problem [27]. This method is suited for various types of combinatorial optimization problems.

Just as in most literature about QAOA, in this project, we consider QAOA applied to Quadratic Unconstrained Binary Optimization (QUBO) problems. However, QAOA can be applied to a wider range of problems. For example, in [28], the authors describe QAOA for the case when the cost function of the optimization problem is an n-th degree polynomial. Nonetheless, in practice, higher-order polynomial unconstrained binary optimization problems are mapped into QUBO problems due to hardware constraints [29].

In this section, we introduce Variational Quantum Algorithms and then discuss arguably the most popular algorithm among them, the QAOA.

2.4.1 Variational Quantum Algorithms

Variational Quantum Algorithms (VQAs) are hybrid algorithms that alternate between a quantum and a classical part. The hybrid loop of a VQA consists of a parametrized

quantum circuit to be run on a quantum computer and a classical optimizer that can update the parameters based on a cost function constructed from outputs of the quantum circuit. In this section, we introduce VQAs to explain the theory behind the QAOA.

Consider the combinatorial optimization problem (1).

The quantum part of VQAs executes a so-called parametrized circuit $U(\theta)$ several times to sample the quantum state. We define it in the following manner.

Definition 2.1. A *parametrized quantum circuit* is defined by a unitary matrix $U(\theta) \in \mathbb{C}^{2^n \times 2^n}$ depending on the parameter $\theta \in \mathbb{R}^d$.

Then, we measure the state $|\psi\rangle = U(\theta)\,|0\rangle^{\otimes n}$ repeatedly to obtain some $x \in \{0,1\}^n$.

Sometimes, we also compute the frequency $p_\theta(x)$ of x in the sampling. That is, the probability of finding x when measuring $|\psi\rangle$ given by $p_\theta(x) = |\langle x|\psi\rangle|^2$. Repeating this process allows us to approximate the probability distribution $\{p_\theta(x) \mid x \in \{0,1\}^n\}$, which can be used to aid the classical optimizer in finding optimal parameters.

The classical part of VQAs consists of varying parameters θ to minimize the energy value $E(\theta) = \langle\psi|\,U(\theta)\,|\psi\rangle$ and finding an approximate $\theta^* = \operatorname{argmin}_\theta E(\theta)$. Often, we cannot directly measure $E(\theta)$ and optimal parameters θ^* are computed through minimization of a so-called *guiding function* $g(\theta)$. The following is a definition of such a function from [28].

Denote by

$$F^*_{\text{quant}} = \left\{ \sum_{s\in F^*} \psi_s\,|s\rangle \;\middle|\; \sum_{s\in F^*} |\psi_s|^2 = 1 \right\}$$

the set of quantum states that are superpositions of optimal solutions of the optimization problem (1).

Definition 2.2. Let $g\colon \mathbb{R}^d \to \mathbb{R}$ be a function and $\mathcal{G}$ the set of its minimizers. We call g a *guiding function* for f with respect to U if g is continuous and

$$\left\{U(\theta)\,|0\rangle^{\otimes n} \;\middle|\; \theta \in \mathcal{G}\right\} \subseteq F^*_{\text{quant}}.$$

This definition means that with a probability 1 measuring the quantum state $U(\theta^*)\,|0\rangle^{\otimes n}$ for $\theta^* \in \mathcal{G}$ outputs a solution to the initial problem $s \in F^*$.

According to [28], a popular choice of guiding function in the literature is

$$g_{\text{mean}}(\theta) = \sum_{x\in\{0,1\}^n} p_\theta(x) f(x).$$

2.4.2 QAOA

In the following, we explain the QAOA for a given QUBO problem of the form (2).

QAOA is an iterative algorithm that alternates between a quantum (QPU) and a classical (CPU) part. Figure 3, adjusted from [29], illustrates how one iteration of the QAOA goes.

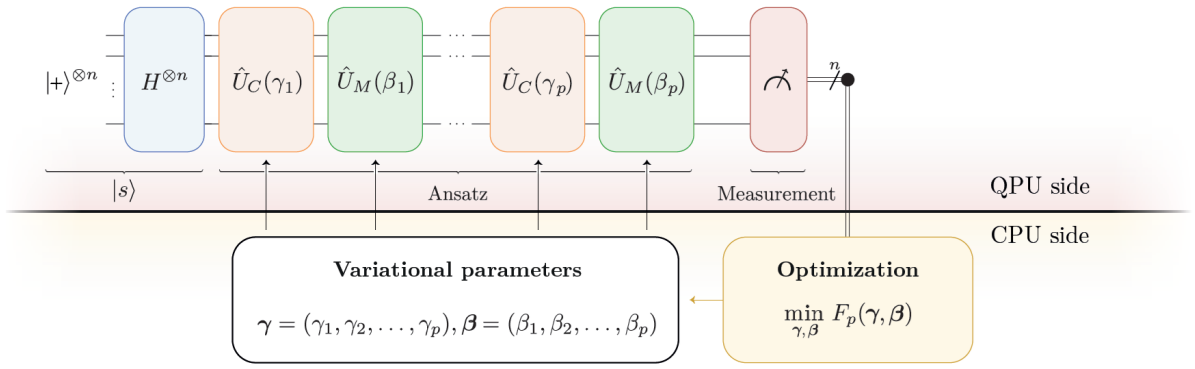


Figure 3: Schematic of the hybrid workflow of QAOA with p layers

The QPU side consists of a parametrized circuit with p layers of alternating cost and mixer layers defined by the unitary gates

$$U_C(\gamma_k) = e^{-i\gamma_k H_C}, \quad U_M(\beta_k) = e^{-i\beta_k H_M}$$

for $k = 1, 2, \ldots, p$, respectively. They are exponentiations of the problem Hamiltonian H_C and a mixer Hamiltonian H_M. The problem Hamiltonian H_C is chosen such that its ground state corresponds to the optimal solution of (2). Concretely, as we derived in the previous section, for QUBO problems, we take

$$H_C = \frac{1}{4}\sum_{i\neq j}^{n} Q_{ij}\sigma_i^z\sigma_j^z - \frac{1}{2}\sum_{i}^{n}\left(Q_{ii} + \sum_{j}^{n} Q_{ij}\right)\sigma_i^z.$$

The mixer Hamiltonian H_M is chosen such that its ground state is easy to find. A common choice is

$$H_M = \sum_{i=1}^{n} \sigma_i^x.$$

Lastly, we take as an initial state the uniform superposition state

$$|s\rangle = |+\rangle^{\otimes n} = \sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{2^n}} |x\rangle .$$

After applying these gates to the initial state $|s\rangle$, we measure it in a computational basis.

Given the parameters $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_p)$ and $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$, this parametrized circuit consisting of $p \geq 1$ layers produces the parameter dependent quantum state

$$|\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle = U_M(\beta_p)U_C(\beta_p)\ldots U_M(\beta_1)U_C(\gamma_1)\,|s\rangle\,.$$

Let F_p be the expectation of H_C in this state:

$$F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}) = \langle \boldsymbol{\gamma}, \boldsymbol{\beta}|\, H_C\, |\boldsymbol{\gamma}, \boldsymbol{\beta}\rangle\,. \tag{5}$$

Usually, we compute through repeated measurements of the final state in a computational basis.

The CPU part of QAOA consists of a classical optimizer that uses this information to find optimal $\boldsymbol{\gamma} = (\gamma_1, \gamma_2, \ldots, \gamma_p)$, and $\boldsymbol{\beta} = (\beta_1, \beta_2, \ldots, \beta_p)$ to run in the next QAOA circuit. This is done by solving the following optimization problem:

$$\min_{\boldsymbol{\gamma}, \boldsymbol{\beta}} F_p(\boldsymbol{\gamma}, \boldsymbol{\beta}).$$

Sometimes, as explained in the previous section on VQAs, to find $(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*)$ that minimize F_p we find $(\boldsymbol{\gamma}^*, \boldsymbol{\beta}^*)$ that minimizes g_p for some guiding function g_p for this problem.

2.4.3 Analysis and performance

In [27], the author showed that QAOA solves optimization problems exactly in the limit $p \to \infty$. However, in practice, we use finite values of p. In fact, QAOA is often used with low values of p. This is because only relatively small-scale quantum computers are available in the current NISQ era. The performance of QAOA for fixed values of p is heuristic as there are no theoretical guarantees.

In this section, we provide a brief literature review about performance guarantees and observed performance results of QAOA. We mainly rely on the review article [29].

The QAOA is considered one of the leading candidate algorithms to achieve a quantum advantage over classical algorithms and QA [29].

It has demonstrated its ability to provide quantum speedup in optimization problems and other domains [29]. Exponential speedup for QAOA over classical methods was proposed in [30] for the MaxCut problem. They estimated that the algorithm with p gates executed on a quantum computer will require $\mathcal{O}(n^2 p)$ gates with runtime $\mathcal{O}(np)$ for a MaxCut problem. In contrast, the most efficient classical approximation algorithm for MaxCut, the Goemans-Williamson, requires a runtime $\mathcal{O}(nm)$ where m is the number of edges of the given graph. Moreover, an exponential speed-up potential of QAOA was suggested for the Minimum Vertex Cover (MVC) problem. In [31], the authors studied QAOA with $2 \leq p \leq 10$ layers and found that it efficiently solves the problem within computational complexity $\mathcal{O}(\text{poly}(k) + \text{poly}(p))$ where k represents the number of iterations in the optimization process. In contrast, competitive decision algorithms

for MVC can find approximate solutions with the computational complexity $\mathcal{O}(2n^2 + n^4)$ where n is the number of vertices in the graph. Moreover, in [32], the authors found an exponential speed-up of QAOA over simulated annealing on a graph of 2D Rydberg atom arrays with up to 289 qubits. It remains an open question whether this advantage can be extended to a more general case. Lastly, in [33], the authors showed that, with a low circuit depth and a short physical duration limit, QAOA can reproduce quadratic speed-up for the unstructured search problem. They further confirmed these results by numerical simulations up to 20 qubits.

While there is literature demonstrating quantum speedup for QAOA for specific problem domains, there are still numerous obstacles to overcome before achieving quantum advantage in a general context and practice. In [29], the authors state that the main hurdles are related to the circuit depth, entanglement, optimization of variational parameters, and noise. For example, in [34], the authors performed realistic simulations of QAOA and concluded that quantum speedup will not be attainable, at least for a representative combinatorial problem, until several hundred qubits are available.

Lastly, various extensions of QAOA have been introduced in the literature to improve runtime performance. The following are a few examples from [29]. The ma-QAOA is QAOA with a multi-angle ansatz with a unique parameter for each element of the cost and mixer Hamiltonians. The ADAPT-QAOA is an iterative version of QAOA with systematic selection of mixers based on a gradient selection. The recursive QAOA is a non-local variant that iteratively reduces the problem size by eliminating qubits. Lastly, FALQON is a variant of QAOA that, instead of a classical optimizer, uses qubit measurements for feedback-based quantum optimization.

3 Audio quantization as a QUBO problem

In this chapter, we introduce the problem of audio quantization. We focus on audio quantization using perception filters and study it through the lens of control theory. Following the work of [1], we show that it is an example of model predictive control for finite-input systems that, under an assumption on the input set, can be rewritten as a QUBO problem.

Throughout this section, we use terms and basic results from control theory. We refer the reader to Appendix B, where we discuss the theory of discrete-time linear systems used in this section.

3.1 Audio quantization

Typically, in order to convert a continuous-time (audio) signal, we first sample the signal to convert it to a discrete set of points. In Figure 4, taken from [35], we depict the process of storing an analog signal digitally.

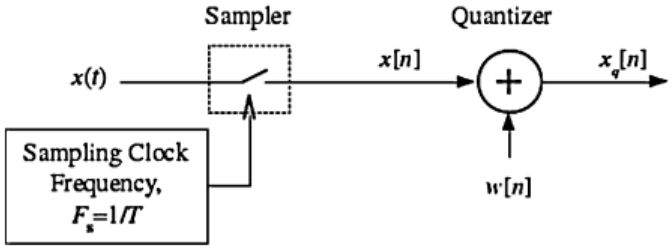


Figure 4: The process of analog to digital conversion with the sampler and quantizer separated

Sampling of a continuous-time signal consists of periodically recording a value of the signal at a given frequency, determining the number of samples per second. For example, consider Figure 5, taken from [35], which depicts periodical sampling of a sinusoidal signal.

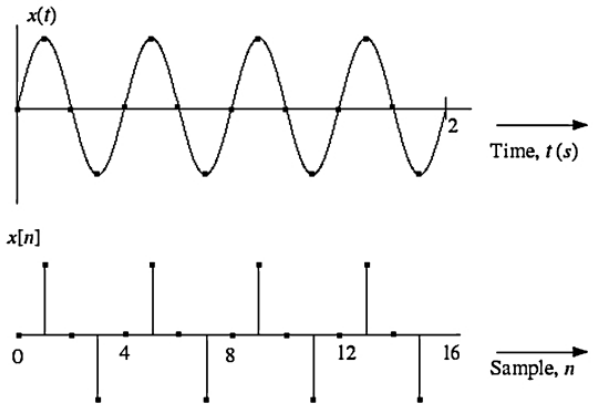


Figure 5: Sinusoidal signal $x(t)$, of frequency 2 Hz

An important result in sampling theory is the Sampling or Nyquist/Shannon sampling theorem that gives us a bound called the Nyquist rate on the sampling rate/ frequency with which we sample. In the following, we provide a mathematical statement of this theorem from [36]. For more details about the Sampling theorem, we refer the reader to the source [36].

Consider a signal $x(t)$ with the Fourier transform

$$X(j\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}\, dt.$$

We call the smallest non-negative frequency f_x (in Hz) such that $X(j\omega) = 0$ for all $|\omega| \geq 2\pi f_x$ the *bandwidth* of the signal. It means that the signal $x(t)$ contains frequencies only up to f_x.

Theorem 3.1. *Let $x(t)$ denote any continuous-time signal having a continuous Fourier transform*

$$X(j\omega) = \int_{-\infty}^{+\infty} x(t)e^{-j\omega t}\, dt.$$

Moreover, let

$$x_d(n) = x(nT),$$

for $n \in \mathbb{Z}$ denote the samples of $x(t)$ at uniform intervals of T seconds. Then $x(t)$ can be exactly reconstructed from its samples $x_d(n)$ if $X(j\omega) = 0$ for all $|\omega| \geq \frac{\pi}{T}$.

Hence, the Sampling theorem states that if a signal $x(t)$ with a bandwidth f_x is sampled at a rate $f_s = \frac{1}{T}$ samples/second, such that $f_s > 2f_x$, then all the information in the continuous-time signal will be retained in the sampled signal, x_d. We call the critical rate $f_s = 2f_x$, the Nyquist rate.

Often, signals are sampled with a high sampling frequency. In order to store the discrete sampled signal digitally, we use a process called quantization. Quantization process consists of converting a finely sampled discrete signal into a finite set of discrete values (or bits). Often, sampling and quantizing of a signal is done in continuous-time, where we quantize a few samples at a time before storing them. Quantization necessarily leads to a loss of information as we convert a signal into a finite set of values. The accuracy of quantization depends on the number of bits used; the larger the number of bits, the more precise the approximation is. Typical values of bits used for audio quantization are 8, 16, 24, or 32 for a fine resolution. Then the number of possible levels is 2 to the power of the number of bits. The difference between the true values and this approximation is called the quantization error.

3.1.1 Audio quantization from a control perspective

In this section, we describe the problem of audio quantization mathematically from a control perspective using [37] and [1].

A vector quantizer is in [37] defined as the following.

Definition 3.1. Given a countable (not necessarily finite) set of non-equal vectors $\mathcal{B} = \{b_1, b_2, \dots\} \subset \mathbb{R}^{n_\mathcal{B}}$, the *nearest neighbour quantizer* is defined as a mapping $q_\mathcal{B} \colon \mathbb{R}^{n_\mathcal{B}} \to \mathcal{B}$ which assigns to each vector $c \in \mathbb{R}^{n_\mathcal{B}}$ the closest element of $\mathcal{B}$ (as measured by the Eucliden norm), i.e., $q_\mathcal{B}(c) = b \in \mathcal{B}$ if and only if c satisfies:

$$\|c - b\| \leq \|c - b_i\|, \quad \forall b_i \in \mathcal{B}.$$

The case $n_\mathcal{B} = 1$ corresponds to a naive solution to the quantization problem. Namely, consider a sequence of sampled scalar audio data, $\{a(t)\},\ t \in \mathbb{N}$, which is either analog or finely quantized. We aim to to quantize it into a signal $\{u(t)\},\ t \in \mathbb{N}$, where $u(t)$ belongs to a finite set

$$\mathbb{U} = \{s_1, \dots, s_m\} \tag{6}$$

for all t. An immediate solution to this problem is at each time $t = k$,

$$u(k) = q_\mathbb{U}(a(k)).$$

However, we can do better than this solution by taking into account that the goal of the quantization process is perceived sound quality. We characterize the human auditory system using perception filters. Then the problem of audio-quantization is as follows:

Given a sequence $\{a(t)\},\ t \in \mathbb{N}$ and a stable perception system filter

$$P(z) = 1 + C(zI - A)^{-1}B, \tag{7}$$

where $A \in \mathbb{R}^{n \times n}$, $B \in \mathbb{R}^{n \times 1}$, and $C \in \mathbb{R}^{1 \times n}$, design an abstract quantizer which minimizes a measure of the filtered error

$$e(t) = P(z)(a(t) - u(t))$$

by choosing the sequence $\{u(t)\}$ where each $u(t) \in \mathbb{U}$. Concretely, given a prediction horizon N at each time t we minimize:

$$H_N(t) = \sum_{k=t}^{t+N-1} (e(k))^2 \tag{8}$$

with respect to

$$\mathbf{u}(t) = \begin{bmatrix} u(t) & u(t+1) & \dots & u(t+N-1) \end{bmatrix}^\top \in \mathbb{U}^N.$$

3.2 Model Predictive Control

In this section, we introduce model predictive control (MPC) and explain how MPC for finite input systems can be turned into QUBO problems. To do so, we follow the work of [1]. Moreover, we show how the audio quantization problem we have described in the previous section is an example of MPC for finite input systems.

We consider discrete-time linear systems of the form

$$\begin{aligned} x(t+1) &= Ax(t) + B_1 u(t) + B_2 a(t) \\ y(t) &= Cx(t) + Du(t) + a(t) \end{aligned} \tag{9}$$

where $A \in \mathbb{R}^{n \times n}$, $B_1 \in \mathbb{R}^{n \times m}$, $B_2 \in \mathbb{R}^{n \times l}$, $C \in \mathbb{R}^{l \times n}$, and $D \in \mathbb{R}^{l \times m}$ are system matrices, $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathcal{U} \subset \mathbb{R}^m$ is the input, $a(t) \in \mathbb{R}^l$ is a given signal, and $y(t) \in \mathbb{R}^l$ is the output of the system. We assume that this system is controllable and observable. In addition, $\mathcal{U}$ is a finite set composed of M vectors such that

$$\mathcal{U} = \{u_i \mid u_i \in \mathbb{R}^m,\ i = 1, \dots, M\}.$$

Where this system differs from a usual control system is that the input $u(t)$ at each time step only takes elements in the set $\mathcal{U}$. To control this system, we design the following evaluation function:

$$H_N(t) = \sum_{k=t}^{t+N-1} y^\top(k) Q y(k) + u^\top(k) R u(k) \tag{10}$$

where $Q \in R^{l \times l}$ and $R \in \mathbb{R}^{m \times m}$ are positive definite symmetric matrices called the weight matrices. Together with N, called the prediction horizon, they are design parameters. We aim to solve the optimal control problem of finding an input sequence

$$\mathbf{u}(t) = \begin{bmatrix} u(t)^\top & u(t+1)^\top & \dots & u(t+N-1)^\top \end{bmatrix}^\top \in \mathbb{R}^{mN}$$

that minimizes the evaluation function (10) while observing the state $x(t)$ at a time t.

3.2.1 MPC to QUBO

In the following, we explain how, under an additional assumption on the input set, one can rewrite the evaluation function (10) as a QUBO evaluation function.

Using the system equations (9), we have that

$$\begin{aligned} y(t+1) &= Cx(t+1) + Du(t+1) + a(t+1) \\ &= CAx(t) + CB_1u(t) + CB_2a(t) + Du(t+1) + a(t+1), \end{aligned}$$

$$\begin{aligned} y(t+2) =& Cx(t+2) + Du(t+2) + a(t+2) \\ =& CAx(t+1) + CB_1u(t+1) + CB_2a(t+1) + Du(t+2) + a(t+2) \\ =& CA^2x(t) + CAB_1u(t) + CAB_2a(t) + CB_1u(t+1) + CB_2a(t+1) \\ &+ Du(t+2) + a(t+2), \end{aligned}$$

and so on. Inductively, we can derive that for any $n \in \mathbb{N}$

$$\begin{aligned} y(t+n) =& CA^nx(t) + CA^{n-1}B_1u(t) + CA^{n-1}B_2a(t) + \dots \\ &+ CB_1u(t+n-1) + CB_2a(t+n-1) + Du(t+n) + a(t+n). \end{aligned}$$

Hence, for

$$\mathbf{y}(t) = \begin{bmatrix} y(t) \\ y(t+1) \\ \vdots \\ y(t+N-1) \end{bmatrix} \in \mathbb{R}^{lN}, \quad \mathbf{a}(t) = \begin{bmatrix} a(t) \\ a(t+1) \\ \vdots \\ a(t+N-1) \end{bmatrix} \in \mathbb{R}^{lN}, \quad \mathbf{A} = \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{N-1} \end{bmatrix} \in \mathbb{R}^{lN \times n},$$

$$\mathbf{B}_1 = \begin{bmatrix} D & 0 & \dots & 0 \\ CB_1 & D & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ CA^{N-2}B & \dots & CB_1 & D \end{bmatrix} \in \mathbb{R}^{lN \times mN}, \quad \mathbf{B}_2 = \begin{bmatrix} I_l & 0 & \dots & 0 \\ CB_2 & I_l & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ CA^{N-2}B_2 & \dots & CB_2 & I_L \end{bmatrix} \in \mathbb{R}^{lN \times lN},$$

we have that

$$\mathbf{y}(t) = \mathbf{A}x(t) + \mathbf{B}_1\mathbf{u}(t) + \mathbf{B}_2\mathbf{a}(t). \tag{11}$$

Notice that, using the vectors $\boldsymbol{y}(t)$ and $\boldsymbol{u}(t)$, we can write the evaluation function (10) as

$$H_N(t) = \mathbf{y}(t)^\top \mathbf{Q}\mathbf{y}(t) + \mathbf{u}(t)^\top \mathbf{R}\mathbf{u}(t), \tag{12}$$

where

$$\mathbf{Q} = \begin{bmatrix} Q & 0 & \dots & 0 \\ 0 & Q & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & Q \end{bmatrix} \in \mathbb{R}^{lN \times lN}, \quad \text{and} \quad \mathbf{R} = \begin{bmatrix} R & 0 & \dots & 0 \\ 0 & R & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \dots & 0 & R \end{bmatrix} \in \mathbb{R}^{mN \times mN}.$$

By substituting (11) into (12) we get that

$$H_N(t) = \mathbf{u}(t)^\top \left(\mathbf{B}_1^\top \mathbf{Q} \mathbf{B_1}\right) \mathbf{u}(t) + 2\left(\mathbf{A}x(t) + \mathbf{B}_2\mathbf{a}(t)\right)^\top \mathbf{Q}\mathbf{B}_1\mathbf{u}(t) + c(t), \tag{13}$$

where $c(t) = \left(\mathbf{A}x(t) + \mathbf{B}_2\mathbf{a}(t)\right)^\top \mathbf{Q}\left(\mathbf{A}x(t) + \mathbf{B}_2\mathbf{a}(t)\right)$ is a constant term independent of $\mathbf{u}(t)$. This is a quadratic form in terms of $\mathbf{u}(t)$. For N, M large, searching the set $\mathcal{U}^N$ to find optimal $\mathbf{u}(t)$ is difficult.

In the following, we convert this optimization problem into a QUBO problem, which can be solved by many algorithms. To do this, we need an additional assumption on the set $\mathcal{U}$.

We assume that $M = 2^L$ for some $L \in \mathbb{N}$ and that all $u \in \mathcal{U}$ satisfy

$$u = \begin{bmatrix} K_1\left(-2^{L-1}b_{1,L} + \sum_{i=1}^{L-1} 2^{L-1}b_{1,i}\right) \\ \vdots \\ K_m\left(-2^{L-1}b_{m,L} + \sum_{i=1}^{L-1} 2^{L-1}b_{m,i}\right) \end{bmatrix} \tag{14}$$

where $b_{i,j} \in \{0,1\}$ for $i = 1, \ldots, m$, $j = 1, \ldots, L$ is a binary variable and $K_i \in \mathbb{R}$ for $i = 1, \ldots, m$ is a scaling constant. This assumption restricts the set $\mathcal{U}$ to a set of discrete values with a regular interval.

The equation (14) can be viewed as the following linear mapping from $\{0,1\}^L$ to $\mathcal{U}$:

$$u = Eb \in \mathcal{U}$$

where

$$E = \begin{bmatrix} E_1 & 0 & \ldots & 0 \\ 0 & E_2 & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & E_m \end{bmatrix} \in \mathbb{R}^{mN \times mLN}, \quad E_i = \begin{bmatrix} K_i & \ldots & K_i 2^{L-2} & -K_i 2^{L-1} \end{bmatrix} \in \mathbb{R}^{1 \times L}$$

for $i = 1, 2, \ldots, m$ and

$$b = \begin{bmatrix} b_{1,1} & \ldots & b_{1,L} & \ldots & b_{m,1} & \ldots & b_{m,L} \end{bmatrix}^\top \in \{0,1\}^{mL}.$$

In addition, for

$$\mathbf{E} = \begin{bmatrix} E & 0 & \ldots & 0 \\ 0 & E & \ddots & \vdots \\ \vdots & \ddots & \ddots & 0 \\ 0 & \ldots & 0 & E \end{bmatrix} \in \mathbb{R}^{mN \times mLN} \quad \text{and} \quad \mathbf{b}(t) = \begin{bmatrix} b(t) \\ \vdots \\ b(t+N_1) \end{bmatrix} \in \{0,1\}^{mLN}$$

the equation (13) can be written as a QUBO as follows

$$H_N(t) = \mathbf{b}(t)^\top \mathbf{J}(t)\mathbf{b}(t) + \mathbf{h}(t)^\top \mathbf{b}(t) + \mathbf{c}(t) \tag{15}$$

where $c(t)$ is a constant independent of $\mathbf{b}(t)$, and

$$\mathbf{J}(t) = \mathbf{E}^\top \left(\mathbf{B}_1^\top \mathbf{Q}\mathbf{B}_1 + \mathbf{R}\right)\mathbf{E}, \quad \mathbf{h}(t) = 2\left(\mathbf{A}x(t) + \mathbf{B}_2\mathbf{a}(t)\right)^\top \mathbf{Q}\mathbf{B}_1\mathbf{E}.$$

Hence, under our assumption on the input set $\mathcal{U}$, minimizing (13) with respect to $\mathbf{u}(t)$ is equivalent to minimizing the QUBO (15) with respect to $\mathbf{b}(t)$. Once (15) is solved, we recover the input via

$$\mathbf{u}(t) = \mathbf{E}\mathbf{b}(t).$$

3.2.2 Audio quantization as MPC

Consider again the problem of audio quantization we have previously defined in Section 3.1.1.

We have that the filtered error, $e(t) = P(z)(a(t) - u(t))$ with the perception filter $P(z)$ (7), satisfies the following equations

$$x(t+1) = Ax(t) + B(a(t) - u(t)), \tag{16}$$
$$e(t) = Cx(t) + a(t) - u(t), \tag{17}$$

where the $x \in \mathbb{R}^n$ is the filter state.

The following is an interpretation from [37] of this filter system. The filter can be regarded as a plant and the signal $u(t)$ as its input, which is restricted to belong to the finite set $\mathbb{U}$. The vector $\mathbf{u}(t)$ should be taken such that the output $P(z)u(t)$ tracks the references $r(t) = P(z)a(t)$ with performance measured in a mean square sense.

We see that if we define the output $y(t) = e(t)$, this filter system (17) corresponds to the MPC system (9) for $B_1 = -B$, $B_2 = B$, and $D = -1$. Moreover, the evaluation function (8) corresponds to the evaluation function (10) for $Q = 1$ and $R = 0$. Then, just as we have previously described, if the input set $\mathbb{U}$ (6) satisfies the assumptions of being a set of discrete values with a regular interval, we can transform this evaluation function into a QUBO evaluation function.

4 Numerical results

In this chapter, we discuss numerical results on the performance of simulated annealing (SA), quantum annealing (QA), and the quantum approximate optimization algorithm (QAOA) applied to the problem of audio quantization.

Throughout this chapter, we study the problem of quantizing an audio signal $a(t)$. We consider uniformly distributed audio data $\{a(t)\}_{t=0}^{T-1} \subset [-1, 1)$, where T is the length of the audio signal corresponding to the number of samples of the signal.

Just as in [1], we consider the following perception filter from [38]

$$P(z) = 1 + z^{-1} \frac{2.245 - 0.664z^{-1}}{1 - 1.335z^{-1} + 0.644z^{-2}} \tag{18}$$

with the sampling frequency 44.1kHz.

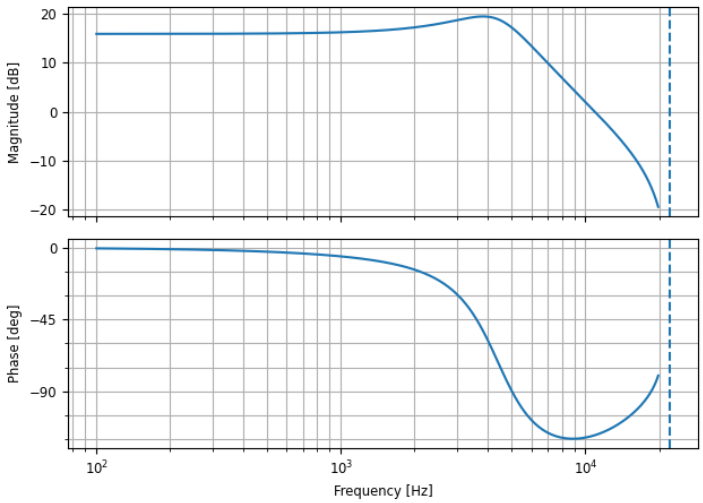


Figure 6: Bode plot of the perception filter

We have that the (controllable) state-space representation [39] of the discrete-time system with the transfer function $P(z)$ is given by

$$\begin{aligned} x(t+1) &= Ax(t) + B\left(a(t) - u(t)\right) \\ y(t) &= Cx(t) + D\left(a(t) - u(t)\right), \end{aligned}$$

where

$$A = \begin{bmatrix} 1.335 & -0.644 \\ 1.0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 1.0 \\ 0 \end{bmatrix}, \quad C = \begin{bmatrix} 2.245 & -0.664 \end{bmatrix}, \quad \text{and} \quad D = 1.$$

At each time t, we aim to minimize the cost function

$$H_N(t) = \sum_{k=t}^{t+N-1} (y(k))^2$$

with respect to $u(t) \in \mathcal{U}$, which we do by solving a corresponding QUBO problem.

The size of the input set $\mathcal{U}$ is related to the number of bits we aim to quantize the signal to. For an L-bit quantization, we will take the input set

$$\mathcal{U} = \left\{-1, \ldots, -2 \cdot 2^{-L+1}, -2^{-L+1}, 0, 2^{-L+1}, 2 \cdot 2^{-L+1}, \ldots (2^{L-1} - 1) \cdot 2^{-L+1}\right\}$$

of 2^L elements.

For example, for $L = 2$, this is the set

$$\mathcal{U} = \{-1, -0.5, 0, 0.5\},$$

and for $L = 3$, the set

$$\mathcal{U} = \{-1, -0.75, -0.5, -0.250, 0.25, 0.5, 0.75\}.$$

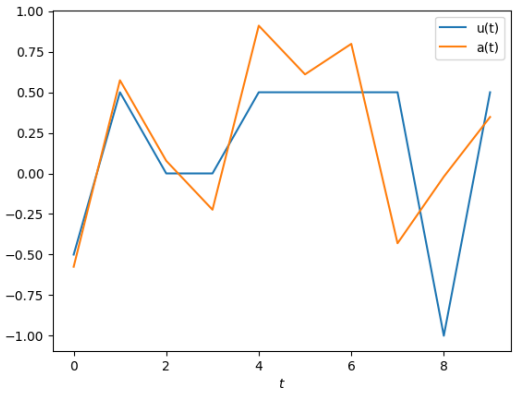
Throughout this chapter, we compare the performance of the methods by comparing the total cost

$$\bar{H} := \sum_{t=0}^{T-1} (y(t))^2.$$

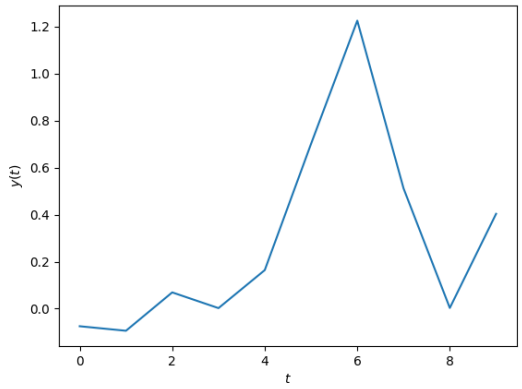
4.1 Parameter tuning

In this section, we tune the parameters of the three solvers we consider. Namely, the QAOA, QA, and SA solvers from the TNO's optimization\qubo package. We consider the prediction horizon $N = 2$, the length of the audio signal $T = 10$ for a problem of audio quantization to L bits.

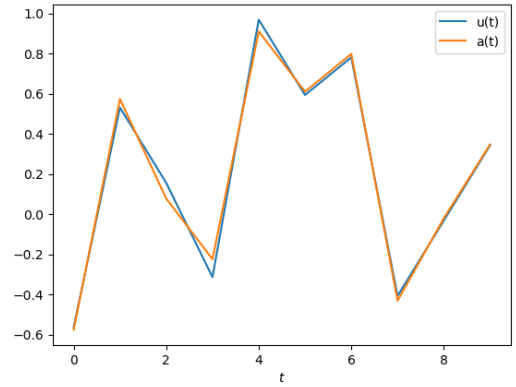
To illustrate how results of such problems look, in Figure 7 we plot the original signal $a(t)$ and the quantized signal $u(t)$ together with the quantization error $y(t)$ for the problem of $L = 2$ and $L = 6$ bit quantization of the same audio signal.



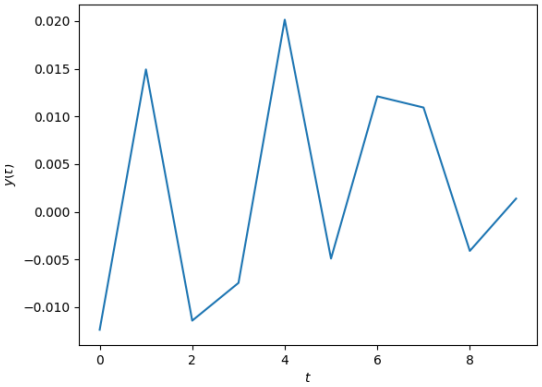
(a) Original and quantized signal, $L = 2$



(b) Quantization error, $L = 2$



(c) Original and quantized signal, $L = 6$



(d) Quantization error, $L = 6$

Figure 7

4.1.1 QAOA parameter tuning

First, we tune the parameters of QAOA. Namely, we tune the choice and the number of iterations/training n_i of a classical optimizer, and the number of layers in the QAOA circuit p. Note that the QAOA solver we use is a simulated QAOA; it fully runs on a classical computer, where it simulates the quantum QAOA circuit. We repeat each of the experiments 5 times to explore how the results vary. We were limited by the computational complexity of simulating quantum circuits, and thus chose a relatively small sample size for the numerical simulations.

First, in Table 1, we compare the performance of the solver applied to the problem of 2-bit quantization for various choices of a classical optimizer. For the rest of the parameters, we will use the default values of the solver. Concretely, $p = 1$ layers, $n_i = 1000$ number of iterations of the optimizer, and 100 shots. Here, shots refer to the number of times we simulated running a prepared quantum state through the QAOA circuit.

optimizer	average total cost	standard deviation
adaptive gradient descent	3.7547	1.7672
adaptive moment estimation	2.4641	0.0000
resilient backpropagation	3.1358	1.4293
stochastic gradient descent	2.7490	0.6371

Table 1: Results for 2-bit quantization with $p = 1$ and varying classical optimizers

We see that adaptive moment estimation, in short ADAM, performs the best. To verify this is also the case for a higher number of layers, we repeat the same experiment with $p = 2$.

optimizer	average total cost	standard deviation
adaptive gradient descent	3.1358	1.4293
adaptive moment estimation	2.4641	0.0000
resilient backpropagation	2.4905	0.0591
stochastic gradient descent	3.1239	1.4729

Table 2: Results for 2-bit quantization with $p = 2$ and varying classical optimizers

From the results in Table 2, we see that indeed ADAM performs the best out of the possibilities of classical optimizers for our use case.

Next, we will apply this choice of optimizer and explore the effect of varying the number of iterations of the optimizer and the number of layers p on the quality of solutions of QAOA.

First, for the problem of 2-bit quantization, we explore the effect of increasing the number of layers p in Table 3. We consider the number of iterations $n_i = 1000$ for the classical optimizer.

p	average total cost value	standard deviation
1	2.4641	0.0000
2	2.4641	0.0000
3	3.5044	1.4916
4	2.5170	0.0724
5	2.4641	0.0000
6	2.4641	0.0000

Table 3: Results for 2-bit quantization with varying number of layers p

We observe that the method performed well already with one layer. Moreover, the results get worse for $p = 3$ and $p = 4$. This can be explained by the fact that with a larger number of layers, the classical optimizer optimizes over a larger number of parameters and thus might need a larger number of iterations before finding optimal values. Moreover, we

only consider a sample size of 5, and the relationship between increasing the number of layers might be clearer with a larger sample size.

To better study the relationship between the number of layers and quality of solutions, we thus focus on $L = 3$ bit quantization. We obtain the following results in Table 4.

p	average total cost value	standard deviation
1	0.5930	0.7945
2	0.2870	0.1974
3	0.3695	0.3090
4	0.3330	0.3268
5	0.1826	0.2126
6	0.1714	0.0998

Table 4: Results for 3-bit quantization with varying number of layers p

We observe that $p = 2$ layers lead to better solutions than just one layer. However, $p = 3$ and $p = 4$ do not lead to better solutions than $p = 2$. For $p = 5$ and $p = 6$, we then observe improvement again.

Note that the standard deviation of these results is relatively high, and with just 5 simulations for each number of layers, we cannot fully determine the effects of increasing the number of layers. However, it seems that for the most part, increasing the number of layers does improve the quality of solutions.

Lastly, we study how the number of iterations of the classical optimizer affects the quality of the results. In Table 5, we fix $p = 1$ and vary n_i for solving the problem of 3-bit quantization.

n_i	average total cost	standard deviation
10	0.6728	0.6565
100	0.3794	0.3097
1000	0.3198	0.3402
2000	0.2411	0.2587
3000	0.2498	0.1799

Table 5: Results for 3-bit quantization with $p = 1$ and varying number iterations n_i

Moreover, since a larger number of layers leads to more variables for the classical optimizer to optimize, we study how the number of iterations affects the quality of solutions for QAOA with $p = 2$ layers in Table 6.

n_i	average total cost	standard deviation
10	0.1393	0.0474
100	0.2361	0.1467
1000	0.2821	0.1628
2000	0.2386	0.1748
3000	0.1132	0.0538

Table 6: Results for 3-bit quantization with $p = 2$ and varying number iterations n_i

We see that a larger number of training, especially for the $p = 1$ case in Table 5, seems to lead to better results. However, the relationship between the number of training and the quality of solutions is not clear for $p = 2$. For example, for just 10 iterations of the optimizer, in Table 6, we find that QAOA performs better than higher amounts of training, except for $n_i = 3000$. Similarly to the number of layers, we cannot conclude decisively what the best parameter is here. This is again due to the fact that we only consider a sample of 5 results, which is quite small.

For the sake of simplicity, we will use the default number of training $n_i = 1000$ of the algorithm from the TNOs package. Moreover, we will consider $p = 1$ and $p = 6$ layers of QAOA for the comparison with other methods.

4.1.2 SA parameter tuning

We tune two parameters for the SA solver from the TNO's package. The first one is the number of reads n_r, and the second one is the number of sweeps n_s. We will repeat 10 times each of the experiments to explore how the results vary.

First, to solve the problem of 2-bit audio quantization, we fix the number of reads to be 1 and vary the number of sweeps n_s in Table 7.

n_s	average total cost	standard deviation
10	3.2372	1.9449
100	3.5911	1.4120
1000	3.0196	1.4404
2000	2.4906	1.1851
3000	3.5368	1.6792

Table 7: Results for 2-bit quantization with varying number of sweeps n_s

We observe that the number of sweeps does not have a clear effect on the quality of the solutions when we consider only one read of the method.

We also study it for a higher-scale problem. In Table 8, we repeat the same experiment, but for the problem of 6-bit quantization.

n_s	average total cost	standard deviation
10	0.6526	0.2445
100	0.4048	0.3103
1000	0.1963	0.1762
2000	0.1523	0.1382
3000	0.1911	0.1639

Table 8: Results for 6-bit quantization with varying number of sweeps n_s

Here, we observe that increasing the number of sweeps improves the quality of the solutions up to $n_s = 2000$. However, for $n_s = 3000$, we observe that the solutions are marginally worse than for $n_s = 2000$. Moreover, we note that the standard deviation in the solutions is rather large in all the experiments we have run. Thus, it is not fully clear whether increasing the number of sweeps significantly improves the quality of solutions.

Now, again for the problem of 2-bit quantization, we fix the number of sweeps to the default value of the solver $n_s = 1000$ and vary the number of reads n_r.

n_r	average total cost	standard deviation
1	2.5658	0.7460
2	2.9557	1.0572
3	2.4773	0.0418
$4 - 10$	2.4641	0.0000

Table 9: Results for 2-bit quantization with varying number of reads n_r

In Table 9, we observe convergence to the same total cost for $n_r \geq 4$. Hence, it is sufficient to consider 4 reads for this problem.

In the literature, common values for the number of reads are between 100 and 1000. However, the problem we tested increasing the number of reads on was relatively small-scale, and hence it is to be expected that a low number of reads will be sufficient. To further study the effect of increasing the number of reads, we consider high-scale problems.

In Table 10, we consider the problem of 6-bit quantization and study how increasing the number of reads affects the quality of solutions.

n_r	average total cost	standard deviation
5	0.01334	0.0206
10	0.0033	0.0022
50	0.0013	0.0002
$100, 500, 1000$	0.0013	0.0000

Table 10: Results for 6-bit quantization with varying number of reads n_r

Here, we see that a larger number of reads is necessary in order to obtain precise solutions. However, we see that 100 reads are already sufficient.

Lastly, we consider a more realistic application of audio quantization where we quantize an audio signal to 16 bits. Then, in Table 11, we observe that 1000 reads might be necessary for high precision.

n_r	average total cost	standard deviation
5	$6.5431 \cdot 10^{-3}$	$4.6580 \cdot 10^{-3}$
10	$1.6737 \cdot 10^{-3}$	$2.1498 \cdot 10^{-3}$
50	$0.0012 \cdot 10^{-6}$	$9.8026 \cdot 10^{-6}$
100	$9.5742 \cdot 10^{-7}$	$9.5345 \cdot 10^{-7}$
500	$1.5291 \cdot 10^{-8}$	$1.2136 \cdot 10^{-8}$
1000	$5.0506 \cdot 10^{-9}$	$4.6031 \cdot 10^{-9}$

Table 11: Results for 16-bit quantization with varying number of reads n_r

For comparison with the other methods, we will thus use SA with the number of sweeps $n_s = 1000$ and the numbers of reads $n_r = 100$ and 1000. Note that for smaller-scale problems, a lower number of reads would have been sufficient.

4.1.3 QA parameter tuning

Lastly, we tune the number of reads n_r parameter of the QA solver. This solver from the TNOs package uses the solver Advantage_system4.1 from D-Wave.

In the literature, common values for n_r are 100 up to 1000. However, for 2 and 3 bit quantization, we had observed convergence of QA to the same total cost values for the number of reads $n_r = 100$ already. Thus, in Table 12 we look at a higher-scale problem of 4-bit quantization, and obtain the following results from running 5 times each of the experiments.

n_r	average total cost	standard deviation
100	0.0312	0.0000
200	0.0308	0.0010
400	0.0312	0.0000
600	0.0312	0.0000
800	0.0312	0.0000
1000	0.0312	0.0000

Table 12: Results for 4-bit quantization with varying number of reads n_r

We observe that the method appears to converge to a value that is not the absolute minimum. For example, for the number of reads $n_r = 200$, one of the 5 simulations found the total cost of 0.0290, which is lower than the rest.

We see that the number of reads $n_r = 100$ is already sufficient for good enough results. This was still a relatively low-scale problem, and hence, it is to be expected that a lower number of reads will be sufficient. In the previous section, we observed that SA performs well with a low number of reads for small-scale problems. Only for larger problems, a high number of reads, like 1000, is necessary for precise results. It is expected that a similar behavior can be found for QA for higher bit quantization problems. Due to our use of QPU time at D-Wave being limited, we did not study the effect of increasing the number of reads for higher-scale problems. In further simulations, just as the authors used in their experiments in [1], we will consider $n_r = 1000$.

4.2 Scaling comparison

In this section, we study how the methods perform with respect to the scaling of the problem size. That is, with respect to increasing the precision of audio quantization. We do so by varying the number of bits L to which we quantize the same audio signal. Just as in the previous section, consider $T = 10$ samples of audio data and the prediction horizon $N = 2$.

In Table 13, we show the average total cost of each method computed from 5 results. For QAOA, we chose the number of layers $p = 1$, the number of training $n_i = 1000$, and 100 shots. Moreover, for QA, we use the number of reads $n_r = 1000$ and for SA, we use the number of sweeps $n_s = 1000$ and the numbers of reads $n_r = 100$ and 1000. Moreover, we add a random sampler (RS) to the comparison to verify whether the methods perform better than random sampling with the same number of reads. That is, the same number of solution attempts from which we pick the best one.

L	QAOA	SA, $n_r = 100$	SA, $n_r = 1000$	QA	RS
2	2.4641	2.4641	2.4641	2.4641	2.4641
3	0.5930	0.0613	0.0613	0.0613	0.0613
4	0.1984	0.0312	0.0312	0.0312	0.0312
5	0.0340	0.0020	0.0020	0.0020	0.0087
6	0.0816	0.0013	0.0013	0.0020	0.0047

Table 13: Total cost values for $2 - 6$ bit audio quantization

We observe that the best performing method is SA, already with $n_r = 100$. The second best is QA. Moreover, QAOA is outperformed by all methods, including the random sampler.

However, if we consider QAOA with 1000 shots instead of 100, it performs better, as one can see in Table 14. Moreover, we have added QAOA with more layers ($p = 6$) to the comparison.

L	QAOA, $p = 1$	QAOA, $p = 6$	SA, $n_r = 100$	SA, $n_r = 1000$	QA	RS
2	2.4641	2.4641	2.4641	2.4641	2.4641	2.4641
3	0.0613	0.0887	0.0613	0.0613	0.0613	0.0613
4	0.0289	0.0382	0.0312	0.0312	0.0312	0.0312
5	0.0183	0.0072	0.0020	0.0020	0.0020	0.0087
6	0.0082	0.0073	0.0013	0.0013	0.0020	0.0047

Table 14: Total cost values for $2 - 6$-bit audio quantization

In particular, for $L = 4$, we find that SA, QA, and the random sampler all find the same total cost in all of the simulations. However, it appears these methods get stuck in a local minimum that QAOA is able to escape. This is because QAOA is able to find a lower total cost than all the other methods. In particular, in one of the simulations, it computed the total cost 0.0167.

For $L = 3$ and $L = 4$, we observe that QAOA with a larger number of layers performs worse than with just one layer. However, theoretically, the results should improve with a larger number of layers. We can explain this by the fact that we have used the same number of iterations of the classical optimizer for both of the methods, and with $p = 6$ layers, there were 6 times as many parameters to optimize. Hence, more iterations of the optimizer might be necessary. Moreover, we only considered a small sample of 5 results for each of the values. Thus, it is possible that with a larger sample size would observe different results.

Nonetheless, for $L = 5$ and $L = 6$, the QAOA is outperformed by all the methods, including the random sampler. This can be partially explained by the fact that for the rest of the methods we considered 1000 reads, whereas for QAOA we did not. Thus, other methods picked the best out of 1000 solutions, whereas the QAOA only took the first one. Due to the time complexity of simulating QAOA, it is not feasible for us to also perform 1000 reads.

Moreover, we note that the random sampler found the same solutions for $L = 2, 3$, and 4. This is to be expected as for smaller-scale problems, it is likely that out of 1000 attempts, it will pick a good solution.

Lastly, for $L = 6$, we see that SA, already with $n_r = 100$, finds a lower total cost than QA. Hence, it performs the best out of all the other methods.

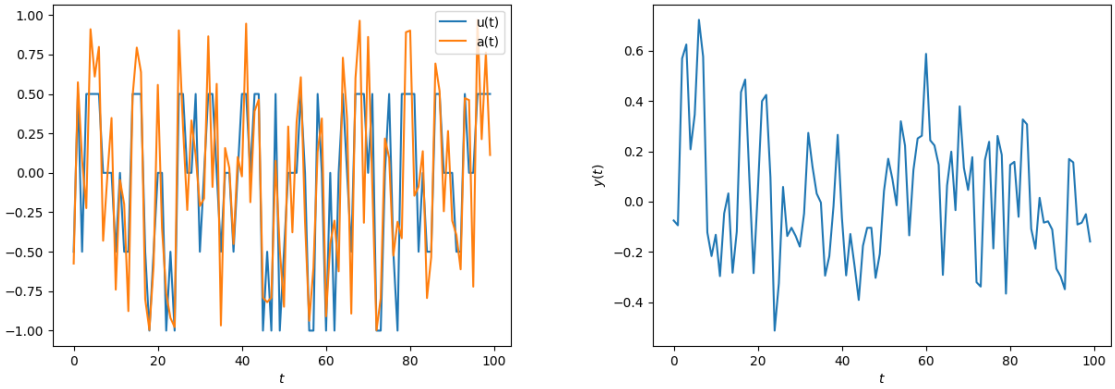
4.3 Comparison with literature example

Lastly, we consider the example studied in [1], where the authors have found that QA outperforms SA. In particular, they have found that the average total cost value of QA is 0.81 times smaller than that of SA (out of 10 numerical simulations). Moreover, they have observed more variance in the results of SA.

Just like in [1], we consider the prediction horizon $N = 10$ and $T = 100$ audio data samples for the problem of 2-bit quantization. We cannot directly reproduce the results from [1] due to the fact that they also took a random audio signal to quantize, and hence we cannot take exactly the same audio signal. Nonetheless, the experiment is almost the same, and the comparison between the methods should lead to similar results.

To illustrate the problem we are solving, in Figure 8 we plot the results for this quantization problem that we have obtained using SA.

Figure 8



(a) Original and quantized signal

(b) Quantization error

We will compare the performance of SA with the parameters $n_s = 1000$, $n_r = 100, 1000$, QA with the number of reads $n_r = 1000$, and QAOA with the parameters $p = 1$, $n_i = 1000$, and 1000 shots. Moreover, we will add to the comparison a random sampler also with the number of reads $n_r = 1000$.

We obtain the following results in Table 15.

method	average total cost	standard deviation
SA, $n_r = 100$	6.7443	0.0953
SA, $n_r = 1000$	6.6539	0.0000
RS, $n_r = 1000$	37.6434	8.3420
QA, $n_r = 1000$	7.6026	-
QAOA, $p = 1$, $n_i = 1000$	-	-

Table 15: Results for the audio quantization problem from [1]

We did 10 simulations for SA and the random sampler, and just 1 for QA. We restrict ourselves to just one simulation for QA due limited QPU usage of annealers from D-Wave. Moreover, we could not simulate QAOA due to the problem size. For $L = 2$ and $N = 10$, the QUBO the algorithms solve is of order 20-bits, which is too large to simulate a QAOA circuit of this size on a personal laptop where these computations were performed.

Unlike the results in [1], we did not observe QA outperforming SA in terms of the quality of solutions. In fact, SA with already $n_r = 100$ found a better solution in all of its results than QA. Moreover, we did not observe a large variance in the solutions of SA.

To add QAOA to the comparison, we now consider the same example but with a lower prediction horizon of $N = 8$. We obtain the following results in Table 15.

method	average total cost	standard deviation
SA, $n_r = 100$	6.4608	0.0998
SA, $n_r = 1000$	6.4293	0.0000
RS, $n_r = 1000$	21.7997	3.360
QA, $n_r = 1000$	6.8218	-
QAOA, $p = 1$, $n_i = 1000$	25.2249	-

Table 16: Results for the audio quantization problem with the prediction horizon $N = 8$

We again performed 10 simulations for SA and the random sampler and only one for QA due to the limited QPU usage of D-Wave annealers. We performed only one simulation for QAOA due to the large computational time of simulating its quantum circuits.

We find that QAOA performs the worst, even worse than the random sampler. Just as for the scaling problem, we note that the comparison was not fully fair, as we allowed only one attempt at a solution for QAOA.

Moreover, for this smaller problem, we find that again, SA performs better than QA even with a lower number of reads, $n_r = 100$. However, the difference between the solutions is less prominent. Both perform significantly better than the random sampler and QAOA.

5 Conclusion and discussion

5.1 Main contributions

In this project, we have expanded on the work of [1]. We have done so in two ways. First, we have added QAOA to their comparison of the performance of SA and QA on the problem of audio quantization. Second, we have studied the performance of these methods with respect to the scaling of the problem.

In Chapter 2, we have introduced combinatorial optimization problems and the three methods we have used in this project to solve them. In order to put our results into context, we have also added an overview of current results about the performance of QA and QAOA, together with what versions of the methods are being studied.

Then, in Chapter 3, we introduced the problem of storing an analog audio signal digitally, and in particular, audio quantization. We have studied it through a control theory perspective using perception filters. Then, just as in [1], we have shown that it is an example of MPC for finite input systems, which under an assumption on the input set, can be reformulated as QUBO problems.

Lastly, in Chapter 4, we have discussed our results where we have expanded on the work of [1]. We have added the QAOA and a random sampler solver to their comparison of QA and SA. For all of these methods, we have used solvers from the TNOs optimization\qubo package. First, we tuned the parameters of these solvers. Then, using the tuned parameters, we have compared the performance of these solvers with respect to the scaling of the problem. We find that for low-scale problems, all methods perform well, even the random sampler. However, for higher-scale problems, both SA and QA performed better. We find that SA performs the best, even with a lower number of reads than QA. Moreover, we find that QAOA performs the worst. One exception to this was the case of 4-bit quantization, where QAOA was able to escape local minima that the other methods could not. Additionally, we have repeated the experiment from [1]. Contrary to their finding, we found that SA outperforms QA in terms of the quality of solutions.

We conclude that, for the use cases that we have studied, the classical algorithm SA outperforms both QA and QAOA.

5.2 Discussion

Our main finding was that in all experiments run, SA performed the best. This is in contrast to [1] where the authors have found that QA offers an advantage. This is despite the fact that our implementation of QA used a newer version of the annealer from D-Wave, the Advantage annealer. It offers significant improvements over the 2000Q used in [1], primarily in terms of problem size capacity, solution quality, and speed.

However, due to our limited QPU usage of the D-Wave annealer, we have limited the number of numerical simulations we run with QA. In our comparison of the performance of the methods on the example from [1], we have run only one experiment with QA, whereas we run 10 experiments with SA. Nonetheless, the result was outside of the standard deviation for the results of SA, and thus, we can conclude that even with more simulations, QA will not perform better.

We suspect that the disparity between our results and the authors' results might be due to parameter tuning of SA, as we have used the same number of reads for QA. The authors have also used the same number of sweeps for SA as we have. Thus, the difference in results might come from the number of reads used for SA. Nevertheless, our choices for the number of reads of SA are commonly found in the literature.

Moreover, the authors compared the computational time of the methods and found that, without accounting for the communication time with the annealer, QA has comparable computational time to SA. For the same example, we have found that, with the same number of reads, SA took 31.6s to compute all 100 steps of the problem, whereas the total QPU time for QA was 21.6s. However, SA with 100 reads found a better solution than QA and took, on average, 3.9s. Therefore, we find that SA not only performs better in terms of accuracy of solutions, but also can be faster if we take 100 reads. We thus conclude that for the audio quantization problems we have considered, SA outperforms all other methods considered.

We were not able to simulate QAOA for this audio quantization problem due to limitations of simulating the QAOA circuit. Thus, in order to compare the methods, we have lowered the prediction horizon to reduce the problem size. Then, we find that again, SA performs the best. Moreover, QAOA did not perform well and found worse solutions than even the random sampler. However, this comparison was not fully fair. Due to the computational complexity of simulating QAOA circuits, we only took one read of the method and could not tune its parameters well. Likely, a well-tuned QAOA solver can perform better than what we have observed.

Moreover, have found that SA also performs the best with respect to scaling of the problem size. However, due to the computational complexity of simulating QAOA, we have restricted our experiments to be relatively small-scale. In order to study how the methods perform with respect to scaling of the problem size better, one should consider a larger prediction horizon and consider audio quantization to a larger number of bits than we have considered. It remains an open question whether QA or QAOA can offer an advantage if we increase the scale of the problem enough.

Lastly, audio quantization is done differently in practice than the approach described in this project. Our approach is not used in practice as audio quantization using perception filters is an NP-hard problem, and there is a need for fast, high-quality solutions. Instead, it is common to first, through perception coding, filter out frequencies less audible to human ears and then quantize the filtered audio samples using nearest neighbour quantizers.

5.3 Extensions and future work

First, as we have mentioned in the discussion section, to better study the performance of the methods with respect to the scaling of the problem size, one should consider a higher prediction horizon and quantization problems with more bits.

Moreover, we were not able to compare QAOA fairly with the rest of the methods. More research is needed on whether QAOA with better-tuned parameters and with a higher number of reads can perform better.

We have only simulated QAOA; an actual QAOA that uses quantum hardware would be able to handle larger-scale problems. However, its solutions would be affected by some noise in the quantum circuits. It would be of interest to repeat our experiments with non-simulated QAOA. We expect it will not perform much better for small-scale experiments, but it might provide an advantage over SA and QA for large-scale problems.

We have found that SA outperformed the other methods. However, its performance could likely be improved further with a more strategic choice of initial state. In our experiments, each read of the algorithm began with a random initial state. It is to be expected that SA can perform better if we were to make a better choice for the initial state. For example, one could take the solution from a previous read of the methods as the new initial state.

One could have improved our results from QA at a relatively low computational cost by post-processing with greedy descent. Then, the optimizer would, through a process of bit-flipping, go over the sample set of QA reads to see if we can find a better solution. Then, it is possible that QA would find the same, if not better, solutions than SA.

Moreover, we have used basic versions of QA and QAOA. As we have explained in Chapter 2, there are many extensions of these methods designed to improve their performance. For example, ma-QAOA, ADAPT-QAOA, recursive QAOA, and FALQON variants of QAOA, and hybrid, improved, and reverse versions of QA. Future research is needed on the efficacy of these and more variants of the methods for the use case of audio quantization.

All the examples of audio quantization problems we have considered were not realistic. In practice, one considers 8, 16, or even 32 bit quantization done continuously on thousands of samples of audio data. Such problems are larger-scale than those considered in this project and require fast solutions. Further research is needed to determine whether quantum computing, potentially through other approaches than those considered in this work, can provide an advantage over classical algorithms for those problems.

Lastly, there are many more applications of MPC for finite input systems. Often, such problems lead to higher-scale optimization problems than audio quantization. We have found that, for this particular application of MPC, quantum computing does not offer an advantage. However, it is possible that for other applications, one could find a quantum speedup using QA or QAOA.

Appendices

A Quantum computing

Quantum computing is an emerging field of technology that leverages the principles of quantum mechanics to perform computations in fundamentally different ways from traditional computers. This section serves as an introduction to quantum mechanics and gate-based quantum computing. Throughout, we rely on the work of [28] and occasionally [6].

A.1 Qubits

In classical computers, we have the computational state 0 or 1. In quantum computers, states can exist in so so-called superposition between these two states. We call such states quantum bits, or in short, qubits.

Definition A.1. We define a *qubit* as

$$|q\rangle = q_0 |0\rangle + q_1 |1\rangle ,$$

where $(q_0, q_1) \in \mathbb{C}^2$ is a pair of complex numbers that satisfies the normalizing condition

$$|q_0|^2 + |q_1|^2 = 1.$$

We say that q_0 and q_1 are the *coordinates* of $|q\rangle$ in the basis $\{|0\rangle , |1\rangle\}$.

When we measure the qubit $|q\rangle = q_0 |0\rangle + q_1 |1\rangle$ we get 0 with probability $|q_0|^2$ and 1 with probability $|q_1|^2$. Matrix representation of the qubit q with respect to the standard basis $\{|0\rangle , |1\rangle\}$ is

$$q = \begin{bmatrix} q_0 \\ q_1 \end{bmatrix} .$$

There are other bases for single qubits, a common example is $\{|+\rangle , |-\rangle\}$, where

$$|+\rangle = \frac{|0\rangle + |1\rangle}{2} \quad \text{and} \quad |-\rangle = \frac{|0\rangle - |1\rangle}{2}.$$

An n-qubit state is defined by 2^n complex numbers that satisfy the normalizing condition and represent the normal decomposition in the canonical basis.

Definition A.2. The *canonical basis* of an n-qubit state is the set

$$\mathcal{CB}_n = \left\{ \bigotimes_{k=1}^{n} \left|i^{(k)}\right\rangle \;\middle|\; (i^{(1)}, \ldots, i^{(n)}) \in \{0,1\}^n \right\},$$

where $i^{(k)}$ represents the k-th qubit.

The canonical basis is the set of all possible combinations of tensor products of n one-qubit basis states, $|0\rangle$ and $|1\rangle$. To illustrate this, consider the following example.

Example 1. The canonical basis of two qubits is as follows.

$$|00\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \quad |01\rangle = \begin{bmatrix}1\\0\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\1\\0\\0\end{bmatrix},$$
$$|10\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}1\\0\end{bmatrix} = \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \quad |11\rangle = \begin{bmatrix}0\\1\end{bmatrix} \otimes \begin{bmatrix}0\\1\end{bmatrix} = \begin{bmatrix}0\\0\\0\\1\end{bmatrix}.$$

Definition A.3. An n-qubit state $|\psi\rangle$ is a normalized linear combination of the basis states in $\mathcal{CB}_n$,

$$|\psi\rangle = \sum_{i\in\{0,1\}^n} \psi_i |i\rangle,$$

where $(\psi_i)_{i\in\{0,1\}^n} \in \mathbb{C}^{2^n}$ are its coordinates, which satisfy the normalizing condition

$$\langle\psi|\psi\rangle = \sum_{i\in\{0,1\}^n} |\psi_i|^2 = 1.$$

The following are definitions of various types of quantum states.

Definition A.4. A quantum state $|\psi\rangle$ is said to be in superposition if $|\psi\rangle = \sum_{i\in\{0,1\}^n} \psi_i |i\rangle$ where there are at least two terms with non-zero coefficients in the sum.

A quantum state is either a product state or an entanglement state.

Definition A.5. An n-qubit state is a *product state* if it is the tensor product of n one-qubit states. In other words, an n-qubit state $|\psi\rangle$ is a *product state* if there exists $2n$ complex coefficients $(q_0^{(j)}, q_1^{(j)})$, $j \in \{1, 2, \ldots, n\}$ such that

$$|\psi\rangle = \bigotimes_{j=1}^{n} \left(q_0^{(j)} |0\rangle + q_1^{(j)} |1\rangle\right)$$

with $|q_0^{(j)}|^2 + |q_1^{(j)}|^2 = 1$ for all $j \in \{1, 2, \ldots, n\}$.

If a qubit is a product state, each state of the qubit can be described independently of the states of the other.

Definition A.6. An n-qubit state $|\psi\rangle = \sum_{i\in\{0,1\}^n} \Psi_i |i\rangle$ is entangled if the numerical values of its coordinates $(\psi_i)_{i\in\{0,1\}^n} \in \mathbb{C}^{2^n}$ admit no solution $(q_0^{(j)}, q_1^{(j)}) \in \mathbb{C}^{2^n}$, $j \in \{1, 2, \ldots, n\}$ such that

$$\begin{cases} \sum_{i\in\{0,1\}^n} \Psi_i |i\rangle = \bigotimes_{j=1}^{n} \left(q_0^{(j)} |0\rangle + q_1^{(j)} |1\rangle \right) \\ |q_0^{(j)}|^2 + |q_1^{(j)}|^2 = 1, \forall j = 1, 2, \ldots, n. \end{cases}$$

It means that operations performed on some coordinate of the entangled state can affect the other coordinates without direct operations on them. Any entangled qubit is in a superposition.

We cannot directly observe quantum states. In order to get information about the state, we need to measure it; however, measurements extract only partial information from it. A single measurement output of a quantum state is a bitstring.

Definition A.7. The measurement $\mathcal{M}$ of an n-qubit state $|\psi\rangle = \sum_{i\in\{0,1\}^n} \Psi_i |i\rangle$ outputs the n-bitstring i with probability $|\psi_i|^2$. After having been measured, the state $|\psi\rangle$ no longer exists; it has been replaced by the state $|i\rangle$.

Hence, measurements of states prepared in the same superposition might lead to different bitstring outputs.

A.2 Quantum circuits

A quantum computer is represented by a quantum circuit containing wires that represent qubits over time and elementary quantum gates that manipulate them.

Quantum gates are modeled by unitary matrices, where a unitary matrix U is such that $U^{-1} = U^\top$. Concretely, a quantum gate that manipulates n-qubit states is a matrix in $\mathbb{C}^{2^n}$ that modifies the 2^n complex coefficients of a quantum state such that they still satisfy the normalizing condition.

Standard quantum gates acting on a single qubit are the following. Pauli matrices σ^x, σ^y, and σ^z:

$$\sigma^x := \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad \sigma^y := \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \quad \sigma^z := \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix},$$

the Hadamard gate H, the phase gate S, and the $\pi/8$ gate T:

$$H := \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}, \quad S := \begin{bmatrix} 1 & 0 \\ 0 & i \end{bmatrix}, \quad T := \begin{bmatrix} 1 & 0 \\ 0 & e^{i\pi/4} \end{bmatrix} = e^{i\pi/8} \begin{bmatrix} e^{-i\pi/8} & 0 \\ 0 & e^{i\pi/8} \end{bmatrix},$$

and the rotation gates (by angle θ over the $x/y/z$ axis):

$$\begin{aligned} R_X(\theta) &:= e^{-i\theta X/2} = \begin{bmatrix} \cos\frac{\theta}{2} & -i\sin\frac{\theta}{2} \\ -i\sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \\ R_Y(\theta) &:= e^{-i\theta Y/2} = \begin{bmatrix} \cos\frac{\theta}{2} & -\sin\frac{\theta}{2} \\ \sin\frac{\theta}{2} & \cos\frac{\theta}{2} \end{bmatrix}, \\ R_Z(\theta) &:= e^{-i\theta Z/2} = \begin{bmatrix} e^{-i\frac{\theta}{2}} & 0 \\ 0 & e^{i\frac{\theta}{2}} \end{bmatrix}. \end{aligned} \tag{19}$$

We have that any single qubit gate is the composition of these rotation gates [6].

Theorem A.1. *Suppose U is a unitary operation on a single qubit. Then there exist real numbers α, β, γ, and δ such that*

$$U = e^{i\alpha} R_Z(\beta) R_Y(\gamma) R_Z(\delta).$$

Moreover, consider the following two-qubit gate:

$$CNOT = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}.$$

We call it the controlled-NOT gate. It acts on the canonical basis such that

$$CNOT\,|a,b\rangle = |a, b \oplus a\rangle\,,$$

for all $a, b \in \{0,1\}$.

We have that a set of gates is said to be *universal* for quantum computing if any unitary operation may be approximated to arbitrary accuracy by a quantum circuit involving those gates only. We have the following result from [6].

Theorem A.2. *The set of rotational gates* (19) *together with the CNOT gate is universal.*

Hence, we can implement any quantum gate using the aforementioned single-qubit gates and the CNOT gate. In fact, on a 2^n-dimensional qubit space, we can exactly implement any quantum gate using $\mathcal{O}(n^2 4^n)$ single qubit and CNOT gates. This is not efficient, a circuit is considered efficient if it can be decomposed in $\mathcal{O}(\text{poly}(n))$ universal quantum gates. In practice, quantum gates are implemented via approximations that use fewer universal gates.

B Discrete-time linear systems

Discrete-time systems are often obtained by sampling continuous-time dynamical systems. They are relevant for computer implementations of controllers and are a natural system description from a data perspective. In this section, we introduce basic concepts in discrete-time linear systems that we use in Chapter 3.

Consider the discrete-time linear system described by the following difference equations

$$\begin{aligned} x(t+1) &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{aligned} \tag{20}$$

which hold for all $t \in \mathbb{Z}_+$. Here $A \in \mathbb{R}^{n\times n}$, $B \in \mathbb{R}^{n\times m}$, $C \in \mathbb{R}^{p\times n}$, and $D \in \mathbb{R}^{p\times m}$ are the system matrices, $x(t) \in \mathbb{R}^n$ is the state, $u(t) \in \mathbb{R}^m$ is the input and $y(t) \in \mathbb{R}^p$ is the output.

We have that for any $t \in \mathbb{Z}_+$,

$$x(t) = A^t x(0) + \sum_{i=0}^{t-1} A^{t-i-1} B u(i),$$

solves the system equations (20). Then, given an initial condition $x(0) = x_0$ and a sequence of inputs $u(0), u(1), \ldots, u(t)$ the resulting output of the system is given by

$$y(t) = CA^t x_0 + \sum_{i=0}^{t} CA^{t-i-1} B u(i) + Du(t).$$

B.1 The transfer function and the z-Transform

We have that the *one-sided z-transform* [39] of a real-valued sequence $\{f(k)\}$ is given by

$$\mathcal{Z}\{f(k)\} := \hat{f}(z) = \sum_{j=0}^{\infty} z^{-j} f(j).$$

An important property following straight from the definition is that

$$\mathcal{Z}\{f(k+1)\} = z\left(\mathcal{Z}\{f(k)\} - f(0)\right) = z\left(\hat{f}(z) - f(0)\right).$$

If we take the z-transform on both sides of the equations (20), we get

$$z\left(\hat{x}(z) - x(0)\right) = A\hat{x}(z) + B\hat{u}(z)$$

and

$$\hat{y}(z) = C\hat{x}(z) + D\hat{u}(z).$$

Thus, we solve that

$$\hat{y}(z) = C\left(zI - A\right)^{-1} zx(0) + \left(C\left(zI - A\right)^{-1} B + D\right)\hat{u}(z).$$

The *transfer function matrix* $\hat{H}(z)$ of the system (20) relates the z-transform of the output $\hat{y}(z)$ to the z-transform of the input $\hat{u}(z)$ under the assumption that $x(0) = 0$. We have that

$$\hat{y}(z) = \hat{H}(z)\hat{u}(z),$$

where

$$\hat{H}(z) = C(zI - A)^{-1}B + D.$$

B.2 Stability

For linear systems, we consider two types of stability, *internal stability* and *input-output stability*. First, we discuss internal stability.

Consider the system

$$x(t+1) = Ax(t) \tag{21}$$

where $A \in \mathbb{R}^{n\times n}$ and $x \in \mathbb{R}^n$.

We define asymptotic stability in the following manner.

Definition B.1. The system (21) is *asymptotically stable* if $x(t) \to 0$ as $t \to \infty$ for all initial states $x(0) \in \mathbb{R}^n$.

We can characterize it in the following manner.

Theorem B.1. *The following statements are equivalent.*

(1) The system (21) *is asymptotically stable.*

(2) The matrix A is Schur, that is, all of its eigenvalues have modulus less than 1*.*

(3) There exists a symmetric and positive definite matrix $P > 0$ such that $P - A^\top PA > 0$.

Consider now the discrete-time system (20) with the transfer matrix
$\hat{H}(z) = C(zI - A)^{-1}B + D$. We define asymptotic stability in the following manner.

Definition B.2. The system (20) is *asymptotically stable* if for $u(t) = 0$ for all $t \in \mathbb{Z}_+$ we have that $y(t) \to 0$ as $t \to \infty$. Moreover, the system (20) is *marginally stable* if there exists a constant $c > 0$ such that for $u(t) = 0$ for all $t \in \mathbb{Z}_+$ we have that $\|y(t)\| \leq c$ for all $t \in \mathbb{Z}_+$.

By the system equations, for a constant zero input, $\lim_{t\to\infty} y(t) = 0$ whenever $\lim_{t\to\infty} x(t) = 0$. Hence, we can also characterize asymptotic stability for this system using Theorem B.1. Moreover, we have that the system (20) is marginally stable if and only if all the eigenvalues of A have modulus less than or equal to 1.

Lastly, we will define a type of input-output stability, called the bounded-input-bounded-output (BIBO) stability.

Definition B.3. The system (20) is *BIBO stable* if there exists a constant $c > 0$ such that if $x(0) = 0$ and $\|u(t)\| \leq 1$ for all $t \in \mathbb{Z}_+$, then $\|y(t)\| \leq c$ for all $t \in \mathbb{Z}_+$.

We can characterize it in the following manner.

Theorem B.2. *The system* (20) *is BIBO stable if and only if all poles of the transfer matrix* $\hat{H}(z)$ *have modulus less than* 1.

Here poles of the transfer matrix $\hat{H}(z) = C(zI - A)^{-1}B + D = \frac{n(z)}{d(z)}$ refer to zeroes of the denominator $d(z)$.

B.3 Controllability and Observability

We define the controllability of the system (20) in the following manner.

Definition B.4. The system (20) is *controllable* if for every pair $x_0, x_1 \in \mathbb{R}^n$, there exists a time $T \in \mathbb{Z}_+$ and a control input $\{u(t)\}_{t=0}^{T-1}$ such that $x(T) = x_1$ for $x(0) = x_0$.

The following theorem characterizes this notion.

Theorem B.3. *The following statements are equivalent:*

(1) The system (20) *is controllable.*

(2) $\text{rank}\begin{bmatrix} B & AB & \dots & A^{n-1}B \end{bmatrix} = n.$

(3) $\text{rank}\begin{bmatrix} A - \lambda I & B \end{bmatrix} = n$ *for all* $\lambda \in \mathbb{C}$.

Moreover, we define observability in the following manner.

Definition B.5. A state $x \in \mathbb{R}^n$ is *unobservable* if, for all $k \in \mathbb{Z}_+$, $CA^k x = 0$. The system (20) is *observable* if $x = 0$ is the only unobservable state.

Similarly to controllability, one can characterize observability in the following manner.

Theorem B.4. *The following statements are equivalent:*

(1) The system (20) *is observable.*

(2) $\mathrm{rank} \begin{bmatrix} C \\ CA \\ \vdots \\ CA^{n-1} \end{bmatrix} = n.$

(3) $\mathrm{rank} \begin{bmatrix} A - \lambda I \\ C \end{bmatrix} = n$ *for all* $\lambda \in \mathbb{C}$.

For a more in depth treatment of discrete-time linear systems, we refer the interested reader to control theory textbooks, for example [39].

References

[1] D. Inoue and H. Yoshida, "Model Predictive Control for Finite Input Systems using the D-Wave Quantum Annealer," *Scientific Reports*, vol. 10, p. 1591, Jan. 2020.

[2] S. E. Venegas-Andraca, W. Cruz-Santos, C. McGeoch, and M. Lanzagorta, "A cross-disciplinary introduction to quantum annealing-based algorithms," *Contemporary Physics*, vol. 59, pp. 174–197, Apr. 2018.

[3] P. Benioff, "The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing machines," *Journal of Statistical Physics*, vol. 22, May 1980.

[4] Y. I. Manin, "Vychislimoe i nevychislimoe (computable and noncomputable), moscow: Sov," 1980.

[5] R. P. Feynman, "Simulating physics with computers," *International Journal of Theoretical Physics*, vol. 21, June 1982.

[6] M. A. Nielsen and I. L. Chuang, *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, Dec. 2010.

[7] P. Shor, "Algorithms for quantum computation: discrete logarithms and factoring," in *Proceedings 35th Annual Symposium on Foundations of Computer Science*, Nov. 1994.

[8] L. K. Grover, "A fast quantum mechanical algorithm for database search," in *Proceedings of the twenty-eighth annual ACM symposium on Theory of Computing*, STOC '96, (New York, NY, USA), Association for Computing Machinery, July 1996.

[9] M. Streif and M. Leib, "Comparison of QAOA with Quantum and Simulated Annealing," Jan. 2019.

[10] S. S. Gill, O. Cetinkaya, S. Marrone, D. Claudino, D. Haunschild, L. Schlote, H. Wu, C. Ottaviani, X. Liu, S. P. Machupalli, K. Kaur, P. Arora, J. Liu, A. Farouk, H. H. Song, S. Uhlig, and K. Ramamohanarao, "Quantum Computing: Vision and Challenges," Apr. 2025.

[11] A. R. Mazumder, A. Sen, and U. Sen, "Benchmarking Metaheuristic-Integrated QAOA against Quantum Annealing," Jan. 2024.

[12] E. Aarts, J. Korst, and W. Michiels, "Simulated Annealing," in *Search Methodologies: Introductory Tutorials in Optimization and Decision Support Techniques* (E. K. Burke and G. Kendall, eds.), pp. 265–285, Boston, MA: Springer US, 2014.

[13] D. Delahaye, S. Chaimatanan, and M. Mongeau, "Simulated Annealing: From Basics to Applications," in *Handbook of Metaheuristics* (M. Gendreau and J.-Y. Potvin, eds.), pp. 1–35, Cham: Springer International Publishing, 2019.

[14] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by Simulated Annealing," *Science*, vol. 220, May 1983.

[15] V. Černý, "Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm," *Journal of Optimization Theory and Applications*, vol. 45, Jan. 1985.

[16] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, "Equation of State Calculations by Fast Computing Machines," *The Journal of Chemical Physics*, vol. 21, June 1953.

[17] A. Das and B. K. Chakrabarti, "Quantum Annealing and Analog Quantum Computation," *Reviews of Modern Physics*, vol. 80, pp. 1061–1081, Sept. 2008. arXiv:0801.2193.

[18] E. Farhi, J. Goldstone, S. Gutmann, J. Lapan, A. Lundgren, and D. Preda, "A Quantum Adiabatic Evolution Algorithm Applied to Random Instances of an NP-Complete Problem," *Science*, vol. 292, pp. 472–475, Apr. 2001. arXiv:quant-ph/0104129.

[19] C. C. McGeoch, *Adiabatic Quantum Computation and Quantum Annealing: Theory and Practice*. Synthesis Lectures on Quantum Computing, Cham: Springer International Publishing, 2014.

[20] J. Roland and N. J. Cerf, "Quantum Search by Local Adiabatic Evolution," *Physical Review A*, vol. 65, p. 042308, Mar. 2002.

[21] S. Teufel, *Adiabatic Perturbation Theory in Quantum Dynamics*, vol. 1821 of *Lecture Notes in Mathematics*. Berlin, Heidelberg: Springer, 2003.

[22] S. Yarkoni, E. Raponi, T. Bäck, and S. Schmitt, "Quantum Annealing for Industry Applications: Introduction and Review," *Reports on Progress in Physics*, vol. 85, p. 104001, Oct. 2022.

[23] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, "Perspectives of quantum annealing: methods and implementations," *Reports on Progress in Physics*, vol. 83, p. 054401, May 2020.

[24] A. Villanueva, P. Najafi, and H. J. Kappen, "Why adiabatic quantum annealing is unlikely to yield speed-up," *Journal of Physics A: Mathematical and Theoretical*, vol. 56, Nov. 2023.

[25] A. Rajak, S. Suzuki, A. Dutta, and B. K. Chakrabarti, "Quantum Annealing: An Overview," *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, vol. 381, Jan. 2023.

[26] L. P. Yulianti and K. Surendro, "Implementation of Quantum Annealing: A Systematic Review," *IEEE Access*, vol. 10, 2022.

[27] E. Farhi, J. Goldstone, and S. Gutmann, "A Quantum Approximate Optimization Algorithm," Nov. 2014.

[28] C. Grange, M. Poss, and E. Bourreau, "An introduction to variational quantum algorithms for combinatorial optimization problems," *4OR*, vol. 21, pp. 363–403, Sept. 2023.

[29] K. Blekos, D. Brand, A. Ceschini, C.-H. Chou, R.-H. Li, K. Pandya, and A. Summer, "A Review on Quantum Approximate Optimization Algorithm and its Variants," *Physics Reports*, vol. 1068, pp. 1–66, June 2024.

[30] G. E. Crooks, "Performance of the Quantum Approximate Optimization Algorithm on the Maximum Cut Problem," Nov. 2018.

[31] Y. J. Zhang, X. D. Mu, X. W. Liu, X. Y. Wang, X. Zhang, K. Li, T. Y. Wu, D. Zhao, and C. Dong, "Applying the quantum approximate optimization algorithm to the minimum vertex cover problem," *Applied Soft Computing*, vol. 118, Mar. 2022.

[32] S. Ebadi, A. Keesling, M. Cain, T. T. Wang, H. Levine, D. Bluvstein, G. Semeghini, A. Omran, J.-G. Liu, R. Samajdar, X.-Z. Luo, B. Nash, X. Gao, B. Barak, E. Farhi, S. Sachdev, N. Gemelke, L. Zhou, S. Choi, H. Pichler, S.-T. Wang, M. Greiner, V. Vuletić, and M. D. Lukin, "Quantum optimization of maximum independent set using Rydberg atom arrays," *Science*, vol. 376, June 2022.

[33] M. Y. Niu, S. Lu, and I. L. Chuang, "Optimizing QAOA: Success Probability and Runtime Dependence on Circuit Depth," May 2019.

[34] G. G. Guerreschi and A. Y. Matsuura, "QAOA for Max-Cut requires hundreds of qubits for quantum speed-up," *Scientific Reports*, vol. 9, p. 6903, May 2019.

[35] S. R. Devasahayam, "Digitization and Discrete Systems: Sampling and Quantization," in *Signals and Systems in Biomedical Engineering: Physiological Systems Modeling and Signal Processing* (S. R. Devasahayam, ed.), pp. 105–133, Singapore: Springer, 2019.

[36] J. O. Smith, *Mathematics of the Discrete Fourier Transform (DFT): With Audio Applications*. Julius Smith, 2007. Google-Books-ID: fTOxS9huzHoC.

[37] D. Quevedo and G. Goodwin, "Audio quantization from a receding horizon control perspective," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 5, pp. 4131–4136 vol.5, June 2003.

[38] R. Wannamaker, "Psychoacoustically Optimal Noise Shaping," *Journal of The Audio Engineering Society*, July 1992.

[39] A. N. M. Panos J. Antsaklis, *A Linear Systems Primer*. Boston, MA: Birkhäuser, 2007.