Toward quantum advantage

Quantum variational algorithms for NP-hard approximations

a thesis presented
by
Austin W. Li
to
The Departments of Physics
and Computer Science

in partial fulfillment of the requirements
for the joint degree of
Bachelor of Arts
in the subject of
Physics and Computer Science

Harvard University
Cambridge, Massachusetts
May 2023

Abstract

Quantum computing is a computing paradigm that promises computational advantage (and in some cases, supremacy) across many domains. Diverse applications across fields (cryptography, complexity theory, generative chemistry, finance) make the development and understanding of quantum algorithms extremely valuable. In this thesis, we consider the application of quantum computing to NP-hard approximation. Through the use of semidefinite relaxations of linear program formulations of the maximum graph bisection problem, and quantum-classical hybrid methods, we show that a novel quantum variational method (HTAAC-QSDP) provides solution quality equivalent to the best classical approximation algorithms. We simulate these hybrid circuits using feed-forward neural networks and determine that the solution quality (cut ratio) achieved is $C_Q/C_{SDP} = 0.987$ for the training graph and $C_Q/C_{SDP} \geq 0.97$ for all other graphs, where C_Q is the cut achieved by the quantum variational method and C_{SDP} is the cut achieved by the classical solver. Our results experimentally verify the results provided in literature and show that quantum variational algorithms may be the path towards achieving near-term quantum advantage.

Contents

1 Introduction **1**

2 Background **4**

2.1 Quantum computation 4

2.2 Semidefinite programs 10

2.3 Variational quantum algorithms 12

3 Simulations and experiments **16**

3.1 Hadamard test objective with approximate amplitude constraints 17

3.2 MaxBisection 19

3.3 Methods 23

4 Results and analysis **27**

4.1 First-order $\mathcal{O}_x$ constraints 27

4.2 Hyperparameter search and generalizability 28

5 Conclusion and future work **33**

A Additional plots and figures **34**

References **42**

Listing of figures

2.1.1 Controlled-NOT circuit and matrix representation. 7
2.1.2 Quantum circuit gate representations. 10

3.0.1 Hadamard test circuit. 17
3.2.1 Quantum implementation of Goemans-Williamson algorithm for MaxBisection with HTAAC-QSDP. 23
3.3.1 Erdös-Rényi graph example with $n = 32, p = 0.8$. 25
3.3.2 Goemans-Williamson algorithm for MaxBisection. 25

4.1.1 Average loss and performance of HTAAC-QSDP on graph 000 for different numbers of Pauli-x string constraints. 29

A.0.1 Average loss and performance of HTAAC-QSDP on graphs 001 through 004 for different numbers of Pauli-x string constraints using the same parameters as Fig. 4.1.1. 35

Acknowledgments

I am extremely grateful to my thesis advisors Professor Susanne Yelin and Dr. Taylor Patti for their guidance and support throughout the thesis research. I am especially thankful to Taylor, whose steadfast guidance and patience throughout the research process — even through weeks with little progress — has shown me what it means to be a physicist, computer scientist, and most importantly, a researcher. I am also very grateful to Susanne for her generosity in taking me on as an advisee on this project and providing rounds of feedback on the manuscript. I would also like to extend thanks to Professors Susanne Yelin, Anurag Anshu, and Boaz Barak for providing thoughtful feedback and comments as my thesis readers.

I would also like to extend thanks to my family and friends for their patience, encouragement, and critique as I worked through this project. I would not have been able to do this without you all. To my friends, thank you for endless encouragement and laughter, for weekend trips and insightful conversations, and for four fantastic undergraduate years. And to my family, thank you for always believing in me. Your support means the world to me.

Chapter 1

Introduction

Quantum computing was first proposed by Richard Feynman in 1982 as a paradigm of computing that could be used to simulate quantum systems [1]. In the decades since Feynman's proposition, quantum computation has developed into a rich field of research, holding promise for a number of applications that has motivated the development of new quantum hardware and algorithms. These quantum algorithms have demonstrated exponential speedups over classical methods in discrete logarithms and factoring [2], solving linear systems of equations [3], and (most importantly for Feynman) simulating quantum systems [4]. The study of quantum information even has applications in drug discovery, generative chemistry [5], and finance [6], where quantum simulations and annealers can be used to accurately model molecules and perform portfolio optimization and credit scoring, respectively.

Modern quantum computing devices often contain 50 to 100 quantum bits (qubits). These devices are referred to as noisy intermediate-scale quantum (NISQ) devices and allow us to achieve "quantum supremacy" [7], outperforming the best classical supercomputer for often contrived mathematical tasks [8, 9]. However, the promise of quantum speedup for practical applications, known as *quantum advantage*, has yet to be realized. Variational quantum algorithms (VQAs) have emerged as a leading strategy to achieve quantum advantage using NISQ devices. VQAs are quantum-classical hybrid

machine learning methods that combine parameterized quantum circuits with classical optimization methods like stochastic gradient descent or neural networks, making use of optimization-based or learning-based approaches to solve problems [10]. Variational methods have found widespread applications in quantum optimization protocols like adiabatic computation [11–13], annealing [14, 15], and the Quantum Approximate Optimization Algorithm (QAOA) [16]. While VQAs may be the key for achieving near-term quantum advantage, they still face challenges in trainability, accuracy, and efficiency [10].

Semidefinite programs (SDPs) are a form of convex optimization where an objective is extremized over the set of symmetric positive semidefinite matrices [17]. While SDPs have a variety of applications spanning computer hardware design and networking [18, 19], we are most interested in using SDPs to approximate difficult combinatorial problems, including NP-hard problems [20, 21]. In many cases, optimization with SDPs have performance guarantees in the form of approximation ratios and represent a compromise between computational complexity and solution quality [22]. In particular, Goemans and Williamson show that SDP relaxations provide performance guarantees for the maximum cut (MaxCut), maximum bisection (MaxBisection), and maximum 2-satisfiability (Max2SAT) problems [20]. Moreover, many quantum SDP algorithms (QSDPs) have been devised over the years [23–27], that provide a quadratic speedup on the number of variables and constraints [28].

Variational methods have recently been proposed for solving QSDPs [29, 30]. One new approach uses Hadamard Test objective functions and Approximate Amplitude Constraints (HTAAC-QSDP), which uses $n + 1$ qubits, a constant number of quantum measurements, and $\mathcal{O}(n^2)$ classical calculations to solve SDPs with $N = 2^n$ variables [31]. Patti et al. demonstrate the feasibility of this novel method by implementing a quantum analog of the classical Goemans-Williamson algorithm for MaxCut [32]. In particular, they find performance and solution quality equivalent to the best classical SDP methods methods for MaxCut. Furthermore, Patti et al. show extensions and applications of the method to additional problems like the MaxBisection problem.

In this thesis, we consider an extension of HTAAC-QSDP for other problems where Goemans-Williamson provides approximation guarantees. We experimentally test the feasibility of HTAAC-QSDP for MaxBisection and show that HTAAC-QSDP indeed provides the same performance guarantee and solution quality as the best known classical SDP solvers.

Chapter 2

Background

In this section, we introduce fundamental concepts of quantum computing, convex optimization, and variational quantum algorithms.

2.1 Quantum computation

2.1.1 Dirac notation

We use bra-ket, or Dirac, notation throughout the thesis. We provide a brief introduction here. The ket is of the form $|v\rangle$, which denotes a vector $\mathbf{v}$ in a complex vector space V. This represents some state of a quantum system. The bra is of the form $\langle v|$ and is the dual vector to $|v\rangle$. A dual vector is a function $f : V \to \mathbb{C}$ that maps each vector to a complex number. We can conveniently think of $|v\rangle$ as a column vector, and $\langle v|$ as the row vector that is the Hermitian conjugate (adjoint) of $|v\rangle$, that is, $\langle v| \equiv |v\rangle^\dagger$.

We let $\langle v|w\rangle$ denote the inner product, $|w\rangle\langle v|$ denote the outer product, and $|vw\rangle = |v\rangle \otimes |w\rangle$ denote the tensor product. One familiar inner product is the dot product or scalar product; in this case, $\langle v|w\rangle$ can be interpreted as the magnitude of the projection of $|w\rangle$ onto $|v\rangle$. An outer product $|w\rangle\langle v|$ is a linear operator from V to W. The action is defined by

$$(|w\rangle\langle v|)|v'\rangle \equiv |w\rangle\langle v|v'\rangle = \langle v|v'\rangle|w\rangle, \tag{2.1}$$

and thus the outer product can be interpreted as an operator that acts on $|v'\rangle$ or the result of multiplying $|w\rangle$ by a scalar $\langle v|v'\rangle$ [33]. The tensor product has the following interesting properties. If V, W are Hilbert spaces with dimension m, n respectively, then $V \otimes W$ is an mn-dimensional vector space with elements that are linear combinations of tensor products $|v\rangle \otimes |w\rangle$ of elements $|v\rangle \in V, |w\rangle \in W$. We often write $|vw\rangle = |v\rangle \otimes |w\rangle$. We note that if $|i\rangle, |j\rangle$ are orthonormal bases for V, W, then $|i\rangle \otimes |j\rangle$ is a basis for $V \otimes W$ [33]. Thus, tensor product states represent states of multiple combined systems.

2.1.2 Quantum bits

The quantum bit, or qubit, is the fundamental unit of quantum information. Unlike classical bits, which can either be in state 0 or 1, qubits can be in any linear combination, or *superposition*, of states. For example, a one-qubit state has a general form

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle, \tag{2.2}$$

for $\alpha, \beta \in \mathbb{C}$ and where $|0\rangle, |1\rangle$ form an orthonormal basis of two-dimensional Hilbert space. The canonical choice of basis is the standard basis. We call $\{|0\rangle, |1\rangle\}$ the computational basis. When we measure a qubit in the computational basis, we get 0 with probability $|\alpha|^2$ and 1 with probability $|\beta|^2$, with $|\alpha|^2 + |\beta|^2 = 1$. This is the normalization condition. Geometrically, then, a one-qubit state is a unit vector in two-dimensional Hilbert space [33].

For multiple qubits, we can turn again to the classical analog. Two classical bits occupy one of the following four states: 00, 01, 10, 11. Likewise, a two-qubit system has four computational basis states, denoted $|00\rangle, |01\rangle, |10\rangle, |11\rangle$. A pair of qubits can exist in any superposition of these four states

$$|\psi\rangle = \alpha_{00}|00\rangle + \alpha_{01}|01\rangle + \alpha_{10}|10\rangle + \alpha_{11}|11\rangle \tag{2.3}$$

with $\sum_{x\in\{0,1\}^2} |\alpha_x|^2 = 1$ and $\alpha_x \in \mathbb{C}$ for $x \in \{0,1\}^2$. In general , we can consider

a system of n qubits with the computational basis states of the system of the form $|x_1 x_2 \ldots x_n\rangle, x_i \in \{0, 1\} \forall i$, and the quantum state of this system specified by 2^n coefficients, or *amplitudes*. Thus, an n-qubit state is a unit vector in 2^n-dimensional Hilbert space [33].

2.1.3 Single qubit gates

Changes to quantum states are described by quantum computation. Analogous to a classical computer, a quantum computer is built from a quantum circuit containing wires and quantum gates. Wires carry information in the circuit and logic gates perform manipulations on the information. The first non-trivial gate is the NOT gate, defined by its truth table $0 \mapsto 1, 1 \mapsto 0$. The quantum NOT gate can be represented by the matrix

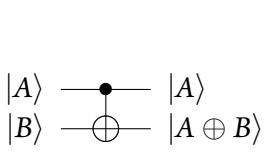
$$X \equiv \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \tag{2.4}$$

where we observe that $X|0\rangle = |1\rangle, X|1\rangle = |0\rangle$, as desired. We note that $X(a|0\rangle + \beta|1\rangle) = a|1\rangle + \beta|0\rangle$, that is, the NOT gate acts *linearly* on quantum states. This is a general (and nice) property of quantum mechanics. We note that any transformation on a quantum state or qubit must preserve the normalization condition that all quantum states have unit norm. Quantum gates, then, must be *unitary*, that is $U^\dagger U = I$ where I is the identity. It turns out that this is the only constraint on quantum gates [33].

The following are other non-trivial single-qubit gates

$$Y \equiv \begin{bmatrix} 0 & i \\ -i & 0 \end{bmatrix}, \qquad Z \equiv \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \qquad H \equiv \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.5}$$

X, Y, Z are known as the Pauli gates and H is the Hadamard gate. We can denote the three Pauli gates using $\sigma_x, \sigma_y, \sigma_z$, respectively. The X, Y, Z gates represent rotations around different axes of a Bloch sphere, a geometrical representation of all possible qubit states. X, Z are particularly useful: X represents a *bit-flip* and Z



$$U_{CN} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 2.1.1: Circuit and matrix representation of controlled-NOT, U_{CN}, written with respect to the amplitudes for $|00\rangle, |01\rangle, |10\rangle, |11\rangle$, in that order. Here, $|A\rangle$ is the control qubit and $|B\rangle$ is the target qubit.

represents a *phase-flip*, leaving $|0\rangle$ unchanged and flipping the sign of $|1\rangle$. H is described as the "square-root NOT" gate and rotates $|0\rangle$ to $\frac{1}{2}(|0\rangle + |1\rangle)$ and $|1\rangle$ to $\frac{1}{2}(|0\rangle - |1\rangle)$ [33]. The Pauli gates are related by $-iXYZ = I$. One can verify that $X^\dagger X = Y^\dagger Y = Z^\dagger Z = H^\dagger H = I$.

2.1.4 Multiple qubit gates

The prototypical multiple qubit gate is the controlled-NOT, or CNOT, gate. Let U_{CN} denote the matrix representation of the gate. The desired behavior of CNOT is as follows:

$$U_{CN}|00\rangle \mapsto |00\rangle, \quad U_{CN}|01\rangle \mapsto |01\rangle, \quad U_{CN}|10\rangle \mapsto |11\rangle, \quad U_{CN}|11\rangle \mapsto |10\rangle. \tag{2.6}$$

We call the first qubit the control qubit and the second qubit the target qubit. We observe that if the control qubit is 0, the target qubit is left alone. If the control qubit is 1, then the target qubit is flipped. In general, any qubit can be the control qubit and any *other* qubit or set of qubits can be the targets. We can conveniently think of the CNOT gate as a generalized XOR. It takes the state $U_{CN}|a, b\rangle \rightarrow |a, a \oplus b\rangle$ where $\oplus$ is bitwise addition modulo two. The circuit and matrix representation of the controlled-NOT, U_{CN}, is given by Fig. 2.1.1. While there are many more quantum gates, the prototypical CNOT and single qubit gates are prototypes for all other gates — any multiple qubit logic gate can be composed from CNOT and single qubit gates. This universality is the quantum

parallel of the universality of the classical NAND gate [33].

A natural generalization of the controlled-NOT gate is a controlled-U gate, where U is any unitary matrix acting on some number n of qubits. Like the controlled-NOT, this family of gates contain a single control qubit and n target qubits. If the control qubit is 0, nothing happens to the target qubits, if the control qubit is 1, then U is applied to the target qubits. Thus, a controlled-NOT is simply a controlled-X gate [33].

2.1.5 Quantum measurement

Recall that measurement of $\alpha|0\rangle + \beta|1\rangle$ in the $\{|0\rangle, |1\rangle\}$ basis yields 0 or 1 with probability $|\alpha|^2, |\beta|^2$, respectively. We can, however, choose to make measurements in a different basis. Consider the basis $|+\rangle \equiv \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle), |-\rangle \equiv \frac{1}{\sqrt{2}}(|0\rangle - |1\rangle)$. We can re-express

$$|\psi\rangle = \alpha|0\rangle + \beta|1\rangle = \alpha\frac{|+\rangle + |-\rangle}{\sqrt{2}} + \beta\frac{|+\rangle - |-\rangle}{\sqrt{2}} = \frac{\alpha+\beta}{\sqrt{2}}|+\rangle + \frac{\alpha-\beta}{\sqrt{2}}|-\rangle \tag{2.7}$$

and thus measurement with respect to this new basis $\{|+\rangle, |-\rangle\}$ yields $+$ with probability $|\alpha+\beta|^2/2$ and $-$ with probability $|\alpha-\beta|^2/2$, with post-measurement states $|+\rangle, |-\rangle$ respectively.

More formally, quantum measurements are *projective measurements*. Following the definitions provided by Chuang and Nielsen [33], a projective measurement is an *observable* M, a Hermitian operator on the state space of the system being observed. The observable has a spectral decomposition

$$M = \sum_m m\Pi_m \tag{2.8}$$

where Π_m is the projector onto the eigenspace of M with eigenvalue m. The possible outcomes of measurement correspond to the eigenvalues m of the

observable. Upon measuring state $|\psi\rangle$, the probability of getting result m is

$$p(m) = \langle\psi|\Pi_m|\psi\rangle = \mathrm{tr}(\Pi_m\rho). \tag{2.9}$$

where $\rho = |\psi\rangle\langle\psi|$ is the density operator. Here, we use the cyclic property of the trace. Given outcome m occurred, the state of the quantum system after measurement is

$$|\psi'\rangle = \frac{\Pi_m|\psi\rangle}{\sqrt{p(m)}}, \tag{2.10}$$

where we see that $|\psi'\rangle$ is renormalized by $1/\sqrt{p(m)}$. Quantum measurement, then, can be described by a collection $\{M_m\}$ of Hermitian measurement operators that satisfy completeness

$$\sum_m M_m^\dagger M_m = I \implies \sum_m p(m) = 1 \tag{2.11}$$

and orthogonality

$$M_m M_{m'} = \delta_{mm'} M_m \tag{2.12}$$

where $\delta_{mm'}$ is the Kronecker delta [33].

One simple example is measurement on a single qubit with two outcomes defined by measurement operators $M_0 = |0\rangle\langle 0|, M_1 = |1\rangle\langle 1|$. Note $M_0^2 = M_0, M_1^2 = M_1$ and $M_0^\dagger M_0 + M_1^\dagger M_1 = M_0 + M_1 = I$. Suppose we measure $|\psi\rangle = a|0\rangle + \beta|1\rangle$. Then the probability of outcome 0 is

$$p(0) = \langle\psi|M_0^\dagger M_0|\psi\rangle = \langle\psi|M_0|\psi\rangle = \langle\psi|0\rangle\langle 0|\psi\rangle = |a|^2 \tag{2.13}$$

and $p(1) = |b|^2$ similarly.

2.1.6 Quantum circuits

Quantum circuits are read left-to-right. Each line in the circuit represents a *wire* in the quantum circuit. The wire may correspond to the passage of time or a physical particle like a photon moving through space; it does not necessarily represent a wire. These wires connect quantum logic gates like the ones discussed in previous sections. Measurement is represented by a "meter" symbol; this operation converts a single qubit state $|\psi\rangle = \alpha|0\rangle + \beta|1\rangle$ into a probabilistic classical bit (which is drawn as a double-line wire). For controlled-U gates, the control qubit is denoted by a line with the black dot [33]. Some examples of quantum circuit diagrams for these gates are given in Fig. 2.1.2. It is convention to assume that state input to circuit is the trivial computational basis state $|0\rangle^{\otimes n}$, where $\otimes n$ denotes taking a tensor product of a state with itself n times [33].

Moreover, we note that quantum circuits are *acyclic* — we do not allow feedback from one part of the circuit to another, and classical operations like FANIN and FANOUT are not allowed in quantum circuits [33].

2.2 Semidefinite programs

Convex optimization techniques have long been applied in quantum computing from quantum error correction [34] to proving quantum complexity results [35]. The subfield of convex optimization we particularly care about is semidefinite

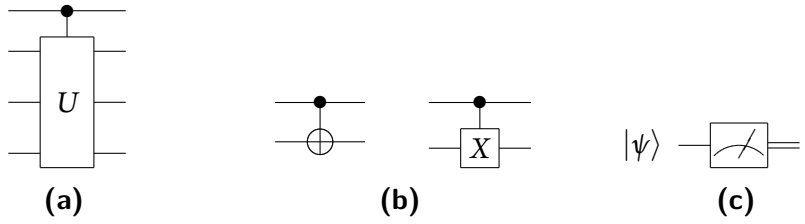


Figure 2.1.2: 2.1.2a Controlled-U gate. 2.1.2b Two representations of the controlled-NOT gate where the top wire contains the target qubit. 2.1.2c Quantum circuit symbol for measurement.

programming. These programming methods are often integer program relaxations, where integer variable objective functions are recast using continuous vector variables [31, 36]. SDPs are useful for solving a broad range of problems, including approximation algorithms for combinatorial optimization [37], control theory [38], and sum of squares [39]. Quantum information problems can be similarly recast using SDPs, including state discrimination [40, 41], upper bounds on quantum channel capacity [42, 43], and self-testing [44]. Generally, SDPs can be solved efficiently in time polynomial in the dimension of the input matrices via classical methods like the interior-point method [29, 45].

We note that although SDPs can be solved efficiently, as the dimension of the input matrices increase, many first-order and second-order methods require gradient calculations at each iteration and require significant computational overhead. Brandão et al. recently proposed a quantum algorithm that provides a quadratic speedup over the classical Arora-Kale algorithm [26, 46]. Following this result, additional quantum algorithms for solving SDPs were developed [23, 25]. Many of these algorithms, however, require fault-tolerant quantum computers to realize these speedups; thus, the design of quantum SDP solvers on NISQ devices is imperative [29].

2.2.1 Formalism

More formally, a semidefinite program is an optimization problem where the goal is to optimize a linear function over the intersection of the positive semidefinite cone with an affine space. SDPs are extensions of linear programs (LPs), where vector inequalities are promoted to matrix inequalities. The standard form of an N-variable, M-constraint semidefinite program is [17]

$$\begin{aligned} \min_{X \in \mathbb{S}^+} \quad & \langle W, X \rangle \\ \text{s.t.} \quad & \langle A_\mu, X \rangle = b_\mu, \qquad \mu \leq M \\ & X \succeq 0 \end{aligned} \tag{2.14}$$

where $W \in \mathbb{S}_N^+$ encodes the optimization problem, $A_\mu \in \mathbb{S}_N^+$ (b_μ) are matrices (scalars) that encode the problem constraints, and $\mathbb{S}_N^+$ denotes the set of $N \times N$ semidefinite matrices. $\langle A, B \rangle$ denotes the trace inner product

$$\langle A, B \rangle = \mathrm{tr} A^\intercal B = \sum_{i,j}^{N} A_{ij} B_{ij}. \tag{2.15}$$

2.2.2 Goemans-Williamson algorithm

The Goemans-Williamson algorithm is a classical NP-hard approximation algorithm. Professors Michel X. Goemans at the Massachussetts Institute of Technology and David P. Williamson at Cornell University significantly advanced the theory of approximation algorithms [20]. Prior to their developments, approximation algorithms largely depended on comparing heuristic solutions to linear program relaxations. Goemans and Williamson's new idea is to use semidefinite programs as relaxations. Briefly, their idea is to recast these problems as SDPs. They then find solutions to these SDPs and use a randomized rounding heuristic to obtain approximations for problems like MaxCut. The randomized rounding heuristic chooses a random hyperplane through the origin, and partitions the vertex set V according to which side of the hyperplane they fall. Solutions obtained by Goemans-Williamson have performance guarantees in the form of approximation ratios, yielding 0.878-approximations for MaxCut and Max2Sat [20]. Later work extends Goemans and Williamson's algorithm to other problems, including Max-k-Cut, MaxBisection [47], and Max2Sat [32].

We use classical cut values obtained by the Goemans-Williamson algorithm to determine the performance and solution quality of the cut values achieved by the quantum variational algorithm.

2.3 Variational quantum algorithms

Variational quantum algorithms are an important class of NISQ algorithms. These are hybrid quantum-classical algorithms characterized by a classical

computer (used for optimization) that can call a quantum subroutine for tasks that it cannot efficiently solve [29]. VQAs have been proposed for certain computational tasks; some well-studied applications include the Variational Quantum Eigensolver (VQE) [10, 30] and QAOA [16].

2.3.1 Formalism

VQAs are designed for solving optimization tasks with an objective or cost function. Without loss of generality, the cost function can be expressed in the form

$$C(\boldsymbol{\theta}) = f(\{\rho_k\}, \{O_k\}, U(\boldsymbol{\theta})) \tag{2.16}$$

where f is some function, $U(\boldsymbol{\theta})$ is a parameterized circuit, $\boldsymbol{\theta}$ are trainable discrete or continuous parameters, $\{\rho_k\}$ are input states from a (possible) training set, $\{O_k\}$ are a set of observables (Hermitian operators). It is useful to express the cost in the form

$$C(\boldsymbol{\theta}) = \sum_k f_k \mathrm{tr}\left[O_k U(\boldsymbol{\theta}) \rho_k U^\dagger(\boldsymbol{\theta})\right] \tag{2.17}$$

for problem-specific functions $\{f_k\}$ [10]. The objective is given by

$$\underset{\theta}{\operatorname{argmin}}\, C(\boldsymbol{\theta}). \tag{2.18}$$

We make the following assumptions about the cost function: it is a faithful encoding of the problem, that is, an extremum corresponds to a solution; smaller cost values indicates better solution quality; the parameters $\boldsymbol{\theta}$ are trainable, that is, they can be efficiently optimized [10].

Another important component of a VQA is the variational ansatz. The parameters $\boldsymbol{\theta}$ can be encoded in a parameterized quantum circuit $U(\boldsymbol{\theta})$ to be used as the variational ansatz. $U(\boldsymbol{\theta})$ is applied to the input state of a quantum circuit, typically $|0\rangle^{\otimes n}$. The form of an ansatz determines the parameters $\boldsymbol{\theta}$, and hence

how they can be trained to minimize $C(\boldsymbol{\theta})$. The specific structure of an ansatz depends on the task, however, there exist problem-agnostic ansatze that can be used when we do not know the structure of the problem a priori [10, 29]. $U(\boldsymbol{\theta})$ can be generically expressed as a product of L sequentially applied unitaries

$$U(\boldsymbol{\theta}) = U_L(\theta_L)\cdots U_2(\theta_2)U_1(\theta_1), \qquad U_\ell(\theta_\ell) = \prod_m e^{-i\theta_m H_m} W_m \tag{2.19}$$

where W_m is an *unparameterized* unitary, H_m is a Hermitian operator, and θ_ℓ is the ℓ-th element in $\boldsymbol{\theta}$ [29]. One such generic variational ansatz is used in our experiments with MaxBisection.

After specifying a cost function and ansatz, we train parameters $\boldsymbol{\theta}$ to solve the optimization problem. It is possible to analytically evaluate the cost function gradient. One training technique that employs partial derivatives is the parameter-shift rule [10]. In this paper, however, we consider stochastic gradient descent (SGD) methods. One SGD method imported from classical machine learning software is `Adam` [48], which adapts the size of steps taken during optimization to allow for more efficient and precise solutions [10]. This is what we use for MaxBisection.

We note that when the dimension of the input is large, the evaluation of expectation values of observables is computationally intractable with classical algorithms [10, 29]. VQAs attempt to circumvent this dimensionality constraint by reducing the optimization to a problem-specific subspace. Specifically, VQAs utilize the parameterized quantum circuit $U(\boldsymbol{\theta})$ to explore a subspace of the input space [29].

2.3.2 Classical parallels

We note that optimizing the parameters $\boldsymbol{\theta}$ of a generic variational ansatz in a quantum circuit is analogous to optimizing the weights of a generic neural network with tunable hyperparameters. Moreover, in VQAs, the protocol for optimization is as follows: each iteration of the (hybrid) feedback loop uses a

quantum computer to efficiently estimate the cost (or gradients); the information is given to a classical computer that uses classical methods to optimize parameters $\boldsymbol{\theta}$. Once a termination condition is met, the VQA outputs an estimate of the solution to the problem.

Because of these properties, we can simulate VQAs on classical computers, where we send an input state $U(\boldsymbol{\theta})|0\rangle^{\otimes n}$, where n is the number of qubits and $U(\boldsymbol{\theta})$ is the variational ansatz, into a feed-forward neural network to optimize the parameters/weights $\boldsymbol{\theta}$. Optimization is done via stochastic gradient descent using `Adam`. The termination condition is defined by the number of epochs or a minimum threshold value on the cost.

Chapter 3

Simulations and experiments

In this section, we introduce HTAAC-QSDP and its application to MaxBisection. We discuss methods, which include data and graph generation, simulation and parameters, and experiments.

3.0.1 The Hadamard test

The Hadamard test is a quantum computing subroutine for arbitrary n-qubit states $|\psi\rangle$ and n-qubit unitaries U. It is a method used to create a random variable with expected value $\mathrm{Re}\langle\psi|U|\psi\rangle$ or $\mathrm{Im}\langle\psi|U|\psi\rangle$ (depending on the initial state configuration). We use the Hadamard test to achieve the latter. To perform a Hadamard test, we first prepare the state $|\psi\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle - i|1\rangle)$, where the $n+1$th qubit is an ancillary qubit and $|\psi\rangle$ is the quantum state of interest. We apply a controlled-U from the $n+1$th qubit to $|\psi\rangle$ to obtain

$$\frac{1}{\sqrt{2}}\left(|\psi\rangle \otimes |0\rangle - iU|\psi\rangle \otimes |1\rangle\right), \tag{3.1}$$

followed by a Hadamard gate on the $n+1$th qubit, yielding

$$\frac{1}{2}\left((I - iU)|\psi\rangle \otimes |0\rangle + (I + iU)|\psi\rangle \otimes |1\rangle\right). \tag{3.2}$$

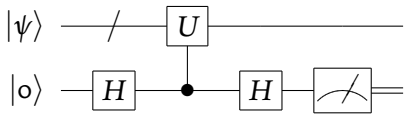


Figure 3.0.1: Circuit for the Hadamard test starting with state $|\psi\rangle \otimes |0\rangle$.

Projective measurement with respect to the $\{|0\rangle, |1\rangle\}$ basis yields $0, 1$ with respective probabilities given by

$$\begin{aligned} p(0) &= \frac{1}{4}\left(2 - i\langle\psi|U|\psi\rangle + i\langle\psi|U^\dagger|\psi\rangle\right), \\ p(1) &= \frac{1}{4}\left(2 + i\langle\psi|U|\psi\rangle - i\langle\psi|U^\dagger|\psi\rangle\right). \end{aligned} \tag{3.3}$$

Calculating the expected value of σ_z, we see that

$$\langle\sigma_z^{n+1}\rangle = -i\frac{\langle\psi|U|\psi\rangle - \langle\psi|U^\dagger|\psi\rangle}{2} = \mathrm{Im}\langle\psi|U|\psi\rangle. \tag{3.4}$$

Thus, the Hadamard test allows the imaginary component of $\langle\psi|U|\psi\rangle$ to be obtained by a single expectation value $\langle\sigma_z^{n+1}\rangle$ where we take an expectation of σ_z on the $n+1$th qubit. To obtain a random variable with expectation $\mathrm{Re}\langle\psi|U|\psi\rangle$, we follow the procedure but start with $|\psi\rangle \otimes \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$. The circuit for this process is given in Fig. 3.0.1.

3.1 Hadamard test objective with approximate amplitude constraints

A novel variational quantum algorithm for solving quantum semidefinite programs (QSDPs) proposed by Patti et al. uses Hadamard Test objective functions and Approximate Amplitude Constraints (HTAAC-QSDP) [31].

The motivation for such a method is as follows: for many worst-case problems (problems with a large number of constraints like MaxCut or MaxBisection), SDP methods may involve the estimation of up to $\mathcal{O}(2^n)$ observables per epoch.

HTAAC-QSDP instead uses $n + 1$ qubits, a constant number of quantum measurements, and $\mathcal{O}(n^2)$ classical calculations to solve SDPs with up to $N = 2^n$ variables. In some cases (high-constraint problems), this provides an exponential speedup in the number of expectation values. We provide a brief overview in the following sections [31].

3.1.1 Hadamard test objective

In quantum analogy to Eq. 2.14, we minimize $\langle W, X \rangle$ over the n-qubit density matrices $\rho = |\psi\rangle\langle\psi|$. Following from Patti et al. [31], we define $|\psi\rangle = U_V|0\rangle^{\otimes n}$ where U_V is the variational quantum circuit and $|0\rangle^{\otimes n}$ the input. This yields an objective function

$$\min_{\rho} \langle W, \rho \rangle = \min \langle \psi | W | \psi \rangle. \tag{3.5}$$

We can encode the objective matrix W as the imaginary part of an n-qubit unitary $U_W = \exp iaW$ where a is a constant scalar. We note that U_W is unitary; one can check that $U_W^\dagger U_W = I$. Following from the above, we can use the Hadamard test to calculate the objective term in the loss function

$$\langle \sigma_z \rangle_W = \mathrm{Im}\langle \psi | U_W | \psi \rangle = \mathrm{Im}\langle U_W, \rho \rangle. \tag{3.6}$$

Thus, we can extremize the N-dimensional objective with the estimation of a single expectation value. We note that the second equality only holds for pure states $|\psi\rangle$.

3.1.2 Approximate amplitude constraints

Approximate amplitude constraints are proposed by Patti et al. to reduce the number of expectation values required to enforce QSDP constraints [31]. When we promote the SDP objective in Eq. 2.14 to an objective of the same form as in Eq. 3.5, our classical constraints must also be promoted to constraints on density

matrices ρ. For MaxCut and MaxBisection, one of these constraints take the form

$$\rho_{ii} = N^{-1}, \quad \forall i \leq N, \tag{3.7}$$

which enforces that all vertices are equally represented in the solution. Enforcing the $M = N = 2^n$ amplitude constraints $\rho_{ii} = N^{-1}$ requires estimation of all Pauli strings with $k \leq n$ Pauli-z operators [31]. Full enforcement requires $\sum_{k=1}^{n} \binom{n}{k} = N - 1 = \mathcal{O}(2^n)$ expectation values. To reduce this large overhead, we consider the set of $\binom{n}{1} + \binom{n}{2} = \mathcal{O}(n^2)$ Pauli strings length $k \leq 2$

$$\begin{aligned} \langle \sigma_z^a, \rho \rangle &= 0, \quad \forall a \leq n \\ \langle \sigma_z^a \sigma_z^b, \rho \rangle &= 0, \quad \forall b \neq a, \quad a, b \leq n \end{aligned} \tag{3.8}$$

as constraints for the n-qubit output state $|\psi\rangle$. This set of constraints approximates (and strongly) enforces the same set of N constraints of Eq. 3.7 [31]. Briefly, the $k = 1$ constraints ensure that each qubit is in equal superposition of $|0\rangle, |1\rangle$ and $k = 2$ constraints prevent 2-qubit correlations that would otherwise satisfy $k = 1$ constraints [31]. One example that Patti et al. provide is the Bell state $|\Phi^+\rangle \equiv \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle)$, which has zero amplitude for states $|01\rangle, |10\rangle$. This state should be disallowed. $|\Phi^+\rangle$ satisfies the $k = 1$ constraints $\langle \sigma_z^1, \rho \rangle = 0, \langle \sigma_z^1, \rho \rangle = 0$, but violates the $k = 2$ constraint $\langle \sigma_z^1 \sigma_z^2, \rho \rangle = 0$.

Thus, HTAAC-QSDP enforces the $M = N = 2^n$ amplitude constraints with the estimation of a polynomial number ($\mathcal{O}(n^2)$) of Pauli-z strings.

3.2 MaxBisection

The maximum graph bisection problem is a well-known graph partition problem. MaxBisection is an extension of the maximum cut problem and is known to be NP-complete via a reduction from MaxCut [49]. The problem is specified by the following. Given a weighted graph $G = (V, E)$ with vertices V and edges E, a *cut* of G is a partition of the vertices into two disjoint subsets $S, V - S$ with

$S \cap V - S = \varnothing$. The size of the cut is the number of edges connecting $S, V - S$; the *maximum cut* is the cut with the greatest weight. We also require that the cut *bisects* G, that is, $|S| = |V - S|$. Thus the maximum bisection problem can be formulated as follows: given such a graph $G = (V, E)$, find this maximum bisection of the graph [49].

3.2.1 Problem specification

We consider an instance of MaxBisection problem on graph $G = (V, E)$ with N vertices. Let $v_i, v_j \in V$ denote two arbitrary vertices. Let W encode the $N(N-1)/2$ non-zero edge weights in entries W_{ij}. Note that $W_{ii} = 0$ because there are no self-edges. The optimization problem has standard form

$$\begin{aligned} \max \quad & \frac{1}{2} \sum_{i<j} W_{ij}(1 - v_i v_j) \\ \text{s.t.} \quad & \sum_j v_j = 0 \\ & v_i = \pm 1, \quad \forall i \leq N, \end{aligned} \tag{3.9}$$

[22]. The SDP relaxation via the Goemans-Williamson algorithm is given by

$$\begin{aligned} \min_{X \in \mathbb{S}^+} \quad & \langle W, X \rangle \\ \text{s.t.} \quad & \sum_{i,j} X_{ij} \leq -N/2 \\ & X_{ii} = 1, \quad \forall i \leq N \end{aligned} \tag{3.10}$$

[47]. As described by Eq. 3.5, we can substitute the classical positive semidefinite matrix X with the quantum density operator ρ. The solution of the SDP is stored in $|\psi\rangle$ where $v_i = \text{sign}(\psi_i)$. Recall evaluation of the objective can be done with the Hadamard test [31]. Patti et al. show that the constraint $X_{ii} = 1$ is equivalent to $\rho_{ii} = 1/2^n = N^{-1}$. Note that $X_{ii} = 1$ ensures that all vertices are in the solution; the equivalent density operator formulation ensures that all states have

the same amplitude $|\psi_i|$, which means that no vertices are disproportionately favored. Moreover, we note that the first constraint of Eq. 3.10 requires half of the variables in X be partitioned equally. Following from the above, we can promote and rewrite Eq. 3.10 as

$$\begin{aligned} \min_{X \in \mathbb{S}^+} \quad & \langle W, \rho \rangle \\ \text{s.t.} \quad & \sum_{i,j} \rho_{ij} \leq -N/2 \\ & \rho_{ii} = N^{-1}, \quad \forall i \leq N \end{aligned} \tag{3.11}$$

where we note that the second constraint can be enforced by the approximate amplitude constraints given in Eq. 3.8. Patti et al. hold that the first constraint of Eq. 3.11 can be enforced with the Pauli-x string constraints

$$\langle \mathcal{O}_x \rangle = 0 \tag{3.12}$$

where $\mathcal{O}_x$ is any Pauli string of σ_x operators [31].

3.2.2 Algorithm

Following Patti et al. [31], we encode the N vertices into n basis vectors where $n = \sup_n \{n : N \leq 2^n\}$. This allows us to encode vertices into a computational basis from which we construct a quantum state $|\psi\rangle$. We use the objective W as a generator of the unitary U_W where

$$U_W = \exp iaW = I + \frac{ia}{1!}W - \frac{a^2}{2!}W^2 - \frac{ia^3}{3!}W^3 + \mathcal{O}(W^4). \tag{3.13}$$

We use the Hadamard test to estimate the objective $\langle \sigma_z^{n+1} \rangle$. We also use a population balancing unitary U_P to address systematic skew due to unequal distribution of graph edges among quantum states. This systematic skew undermines Eq. 3.7 [31]. We implement this on $|\psi\rangle$ via a second Hadamard test, which adds the loss term $\langle \sigma_z^{n+1} \rangle_P = \text{Im} \langle \psi | U_P | \psi \rangle$. We define the population

balancing unitary

$$U_P \equiv \exp i\beta P, \qquad P_{ii} = -\left(P_{\max} - \sum_j |w_{ij}|\right) \tag{3.14}$$

where β is an adjustable hyperparameter and $P_{\max}$ is the maximum magnitude of edge weights for any vertex [31]. We note that P is diagonal. U_P balances state populations by promoting states that are less represented or absent from the objective. In our simulations, the strength of our population balancing term is regularized via a hyperparameter r.

Thus, combining the Hadamard test objective, approximate amplitude constraints in Eq. 3.7, and the Pauli-x string constraint in Eq. 3.12, we can use gradient descent-based penalty methods to find a solution to the following loss function [50], given by

$$\begin{aligned}\mathcal{L}(t) = \langle \sigma_z^{n+1} \rangle_{W,t} + \frac{1}{r}\langle \sigma_z^{n+1} \rangle_{P,t} \\ + \lambda_Z \left[\sum_j \langle \sigma_z^j, \rho_t \rangle^2 + \sum_{k \neq j} \langle \sigma_z^j \sigma_z^k, \rho_t \rangle^2\right] + \lambda_X \left[\sum_j \langle \sigma_x^j, \rho_t \rangle^2\right].\end{aligned} \tag{3.15}$$

Here, we impose constraints by encoding them as penalty terms to our objective.

An algorithm outline is as follows. We prepare a generic variational ansatz U_V and initial state $|\psi\rangle = U_V(t)|0\rangle^{\otimes n}$. At each time step t, we prepare $\rho_t = |\psi_t\rangle\langle\psi_t|$ on a variational quantum computer, conduct Hadamard tests and measurements, and optimize our variational ansatz parameters $\boldsymbol{\theta}$ via backpropagation in a classical feed-forward neural network. We note r is a regularization hyperparameter that regularizes the strength of the population balancing term. λ_Z, λ_X are the penalty hyperparameters on the Pauli-z (approximate amplitude) and Pauli-x (bit-flip) constraints, respectively. For simplicity, we have chosen time-constant λ_Z, λ_X values, although we note that λ_Z, λ_X can also be time-dependent. The pseudocode for the algorithm is given in Alg. 3.2.1.

Data: Optimization matrix W, Hadamard coefficients a, β, learning rate η, regularization r, circuit depth d, penalties λ_Z, λ_X, number of epochs T.

Initialize variational quantum circuit U_V

for t **in range:** T

do

$|\psi_t\rangle \otimes |0\rangle = U_V(t)|0\rangle^{\otimes n} \otimes |0\rangle$

Hadamard test (phase a) $U_W \to \langle \sigma_z^{n+1} \rangle_{W,t}$

Hadamard test (phase β) $U_P \to \langle \sigma_z^{n+1} \rangle_{P,t}$

Measure $\langle \sigma_z^j \rho_t \rangle^2, \langle \sigma_z^j \sigma_z^k, \rho_t \rangle^2, \langle \mathcal{O}_x \rangle^2$

Calculate $\mathcal{L}(t) = \langle \sigma_z^{n+1} \rangle_{W,t} + \frac{1}{r} \langle \sigma_z^{n+1} \rangle_{P,t} +$
$\lambda_Z \left[\sum_j \langle \sigma_z^j, \rho_t \rangle^2 + \sum_{k \neq j} \langle \sigma_z^j \sigma_z^k, \rho_t \rangle^2 \right] + \lambda_X \left[\sum_j \langle \sigma_x^j, \rho_t \rangle^2 \right]$

Backpropagate $\eta \nabla \mathcal{L}(t) : U_V(t) \to U_V(t+1)$ and update parameters $\boldsymbol{\theta}$

end

Result: SDP MaxBisection vertex solution $|\psi_T\rangle, \text{sign}(|\psi_T\rangle) = [\nu_1, \cdots, \nu_n]^\intercal$.

Figure 3.2.1: Quantum implementation of Goemans-Williamson algorithm for MaxBisection with HTAAC-QSDP.

We note that Eq. 3.15 indicates we only use Pauli-x strings that only contain one σ_x. We later empirically justify this decision. More specifically, we show that in fact the addition of longer Pauli-x strings does not affect solution quality, but rather causes stronger violation of our bit-flip constraints $\langle \mathcal{O}_X \rangle = 0$.

3.3 Methods

We show the feasibility and solution quality of HTAAC-QSDP on MaxBisection by comparing the cut size achieved by HTAAC-QSDP to cuts produced by a classical SDP solver. Let C_Q be the cut achieved by HTAAC-QSDP and C_{SDP} be the cut achieved by the classical SDP method (Goemans-Williamson). We want to show that $C_Q / C_{\text{SDP}} \approx 1$, that is, the QSDP and classical solvers converge to the same cut size. By doing this, we show the two methods have equivalent solution quality.

3.3.1 Data and graph selection

A repository of graphs for a variety of graph partitioning problems was generously provided by the Graph Partitioning Archive, which is compiled and maintained by Professor Chris Walshaw at the University of Greenwich [51]. The repository contains the best known performance values and associated algorithms for 34 graphs, including SDP methods. Initial simulations were conducted on the two smallest graphs, but graph size ($\mathcal{O}(10^3)$ vertices and $\mathcal{O}(10^5)$ edges) and computational limitations made large-scale simulations time-inefficient.

We instead generate Erdös-Rényi, or binomial, graphs using the `NetworkX` package [52]. We would like to thank Robin Alexandra Brown, a Ph.D. candidate at Stanford University, for her help with graph generation. More specifically, we generate graphs $G(n, p)$ with n nodes, where nodes are connected with probability p [53, 54]. The expected number of edges in binomial graphs is $\langle |E| \rangle = \binom{n}{2} p$. We choose $n = 32, p = 0.8$ as our parameters and generate ten graphs, labeled `000` to `009`. Given these parameters, we expect (on average) each graph to contain 396.8 edges. One example of such a graph generated with our desired parameters is given by Fig. 3.3.1. We then produce SDP solutions on the graphs with Goemans-Williamson using the `CVXPY` package [55, 56]. We compare our HTAAC-QSDP cut values to these classical cut values. Pseudocode for generating classical SDP solutions is provided in Alg. 3.3.2. We use graph `000` for training and graphs `001` to `009` for testing.

3.3.2 Simulation specification and parameters

We run a simulation of Alg. 3.2.1 using the `Tensorly-Quantum` simulator [57, 58]. Our chosen circuit ansatz U_V is comprised of d repetitions of two variationally parameterized Y gates interleaved with CNOT gates. We let d denote the circuit depth or the number of gate repetitions. The CNOT gates alternate between odd-even and even-odd qubit control. This is the same ansatz used by Patti et al. [31]. Optimization by stochastic gradient descent is

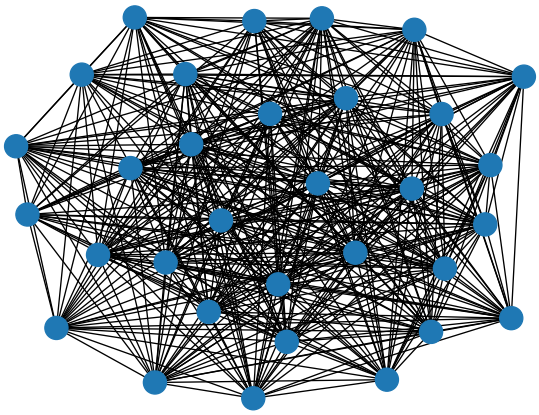


Figure 3.3.1: Example Erdös-Rényi graph with $n = 32, p = 0.8$ generated using the `NetworkX` package. These are the types of graphs we use to test HTAAC-QSDP.

Data: Number of nodes n, edge probability p, number of trials N
Generate Erdös-Rényi graph $G(n, p)$
Solve classical SDP for MaxBisection in `CVXPY` to obtain SDP solution
for i **in range:** N
do
    Perform randomized rounding to generate cut on SDP solution, store cut size
    Balance cut to produce bisection cut, store cut size
end
Result: MaxBisection cut value and graph $G(n, p)$.

Figure 3.3.2: Goemans-Williamson algorithm for MaxBisection.

conducted using the `Adam` optimizer [48].

All graph and algorithm hyperparameters (α, β, learning rate η, U_V circuit depth d, penalization base values c_b, regularization r) are allowed to vary freely with the exception of the penalty parameters λ_Z, λ_X. These are defined by

$$\lambda_i \equiv \frac{c_b \alpha}{N_i}, \qquad i \in \{Z, X\} \tag{3.16}$$

where c_b is a hyperparameter that defines the coefficient base value for penalties

and N_Z, N_X are the number of σ_z, σ_x string constraints, respectively. We note that we require $2^n N_Z$ *and* at most $2^n N_X$ constraints to fully enforce all constraints, however, we show that we can approximate these constraints using approximate amplitude constraints *and* first-order bit-flip constraints, respectively.

3.3.3 Hyperparameter search

Because we have many parameters that are allowed to vary, we perform a manual hyperparameter search to find the best-performing parameters for our graphs. We conduct hyperparameter search on graph `000` and test the performance of optimal parameters on graphs `001` through `009`.

3.3.4 Optimization

Early simulations of HTAAC-QSDP for MaxBisection required many hours. To speed up simulation time and reduce computational overhead, the objective and constraints are vectorized. This decreases computation time by leveraging the implementation of `NumPy` operations in C [59]. Moreover, we utilize sparse tensors for the objective U_W. Sparse tensors enable efficient storage and processing of tensors that contain many empty values. We use `PyTorch` implementations of sparse matrices [60]. With these optimizations, simulations experience speedups that are many orders of magnitude faster than the unoptimized procedure.

Chapter 4

Results and analysis

In this chapter, we discuss main results and analysis.

4.1 First-order $\mathcal{O}_x$ constraints

We run Alg. 3.2.1 on subsets of all 31 $\langle \mathcal{O}_x \rangle$ constraints on graph 000. Recall that we run experiments with $n = 5$ qubits (graphs generated with $2^n = 32$ vertices). For these experiments, we set our hyperparameters to

$$\eta = 0.005, \qquad c_b = 100, \qquad r = 1.2, \qquad d = 120, \qquad T = 150 \tag{4.1}$$

where r is U_P regularization and we average over 10 reps per trial. These hyperparameters are similar to ones used in HTAAC-QSDP experiments with MaxCut [31]. The values we choose are a result of preliminary testing, where we sweep the parameters across a small range of values to achieve the best cut values for graph 000.

We run trials on the first 5, 15, 25, 30, 31 constraints, ordered by the number of σ_x gates that each Pauli-x string contains. For example, the first $\binom{5}{1}$ Pauli-x strings contain a single σ_x and the next $\binom{5}{2}$ Pauli-x strings contain two σ_x operators. Expectation values are tabulated in Tab. 4.1.1. We observe that as we increase the number of included operators, longer strings are more strongly enforced (recall

we want $\langle \mathcal{O}_x \rangle = 0$) and shorter strings are more strongly violated.

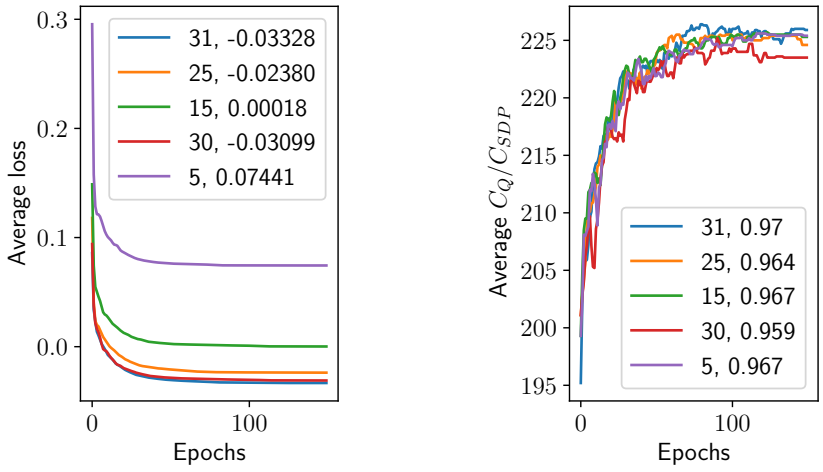
We also look at the loss and cut size for each set of constraints on graph 000. Recall C_Q denotes the cut size achieved by HTAAC-QSDP and C_{SDP} denotes the cut achieved by the classical Goemans-Williamson algorithm. We have a plot of the loss and cut ratio C_Q/C_{SDP} given by Fig. 4.1.1. We see that in general, the inclusion of more Pauli-x strings results in a marginal decrease in the loss; yet the cut ratio C_Q/C_{SDP} converges to 0.96 for all subsets of constraints. Thus, the performance of HTAAC-QSDP is independent of the number of included strings/constraints. We also conduct these tests for graphs 001 through 004. The loss and cut plots are given by Fig. A.0.1 in Appendix A. That decreasing the number of constraints decreases computational overhead *and* does not affect solution quality motivates us to include only the $\binom{n}{1} = \mathcal{O}(n)$ Pauli-x strings that contain one σ_x. These are **first-order bit-flip constraints**, which are similar to the approximate amplitude constraints used to approximately enforce $\rho_{ii} = N^{-1}$.

4.2 Hyperparameter search and generalizability

We conduct a manual hyperparameter search to find the optimal parameters for: learning rate η, gate repetitions (circuit depth) d, the unitary phase a, coefficient base c_b, and U_P regularization r. We perform this search on graph 000 and use first-order bit-flip constraints, using Pauli-x string constraints containing exactly one σ_x operator, for $T = 150$ epochs.

The first search includes a wider range of values, followed by a second, more restricted, search based on the performance of the first search. We search over $\mathcal{O}(10^4)$ different combinations of parameters across the two searches. The best cut ratios for all permutations of parameters is given by $C_Q/C_{SDP} = 0.987$. We tabulate results of the best performing parameters in Tab. 4.2.1.

We test the parameters in Tab. 4.2.1 on all graphs 001 through 009 for three trials each. The best-performing parameters, average cut values and ratios, and performance are tabulated in Tab. 4.2.2. We expect the cut ratio to be weakly



(a) **(b)**

Figure 4.1.1: Plots generated on graph 000 using a learning rate $\eta = 0.005$, coefficient base $c_b = 100$, U_P regularization $r = 1.2$, $T = 150$, a circuit depth of 120 and 10 repetitions per trial. 4.1.1a The average loss of 10 repetitions after 150 epochs using different number of Pauli-x string constraints. The legend is in the form (c, v), where c is the number of constraints and v is the (rounded) value at time T. We note that average losses can be negative because all terms in the loss function are not strictly positive. 4.1.1b. The average cut ratio C_Q/C_{SDP} of 10 repetitions after 150 epochs using different number of Pauli-x string constraints. The legend is in the same form as in Fig. 4.1.1a. We see that the average cut ratio converges to some maximum value.

smaller than the cut ratio for graph 000 because the optimal parameters $\boldsymbol{\theta}$ are problem-specific. We find, however, that for most of the graphs, solution quality exceeds 0.97, which demonstrates that this method generalizes well to graphs outside the training set. The top five performing sets of parameters for each graph are tabulated in Tab. A.0.1 in Appendix A.

Looking at these plots, we demonstrate that HTAAC-QSDP using first-order bit-flip constraints indeed produces approximate solutions/cut values with solution quality near or equivalent to the best-known classical SDP solvers for MaxBisection.

	Total number of constraints				
i	5	15	25	30	31
0	0.100505	0.139088	0.169367	0.172688	0.178030
1	0.100332	0.139688	0.169438	0.172601	0.173783
2	0.107630	0.150811	0.167124	0.181454	0.185950
3	0.098748	0.130857	0.155605	0.159990	0.162059
4	0.101498	0.132806	0.160310	0.163272	0.168414
5		0.013812	0.026385	0.026127	0.027051
6		0.016460	0.023906	0.027786	0.030617
7		0.014324	0.023942	0.024878	0.026348
8		0.014538	0.022428	0.024895	0.026268
9		0.014854	0.020206	0.025320	0.025725
10		0.015090	0.025445	0.026211	0.027647
11		0.012768	0.022046	0.023208	0.025115
12		0.016287	0.022063	0.025470	0.027178
13		0.017697	0.027633	0.028944	0.031180
14		0.014333	0.023956	0.024749	0.026587
15			0.010557	0.009585	0.010803
16			0.002158	0.002708	0.003311
17			0.004382	0.005106	0.004740
18			0.002164	0.001709	0.002495
19			0.002551	0.004663	0.003833
20			0.002534	0.002653	0.003088
21			0.002401	0.003039	0.002397
22			0.000894	0.002039	0.001676
23			0.006467	0.009419	0.011658
24			0.000700	0.001218	0.000932
25				0.000360	0.000510
26				0.000379	0.000402
27				0.000854	0.001160
28				0.000011	0.000004
29				0.000682	0.000502
30					0.000002

Table 4.1.1: $\langle \mathcal{O}_x \rangle$ values of the ith Pauli-x string as we increase the total number of Pauli-x strings. A larger $\langle \mathcal{O}_X \rangle$ value indicates greater violation of the constraint $\langle \mathcal{O}_x \rangle = 0$, which enforces vertex bisection. We order strings by the number of σ_x gates in the Pauli-x string. For example, the first $\binom{5}{1}$ constraints contain all Pauli-x strings with **one** σ_x; the next $\binom{5}{2}$ constraints contain all strings that contain **two** σ_x gates.

Test	c_b	η	α	r	d	C_Q/C_{SDP}
0	150	0.01	0.05	1.0	40	0.987
1	150	0.01	0.05	1.5	60	0.987
2	100	0.01	1.0	1.0	40	0.987
3	100	0.005	0.01	1.0	20	0.987
4	100	0.005	0.05	2.0	60	0.987
5	100	0.005	0.05	1.5	60	0.987
6	200	0.01	0.01	2.0	60	0.987
7	150	0.005	0.5	1.0	60	0.987
8	100	0.005	0.01	2.0	60	0.987
9	100	0.01	0.01	1.5	40	0.987
10	100	0.005	0.5	1.0	40	0.987
11	200	0.01	0.01	1.0	60	0.987
12	100	0.005	0.1	2.0	60	0.987
13	100	0.005	1.0	2.0	60	0.987
14	100	0.0025	0.05	1.5	20	0.987
15	100	0.0025	0.5	1.5	60	0.987
16	100	0.01	1.0	1.0	60	0.987
17	100	0.0025	1.0	1.0	40	0.987
18	100	0.01	0.05	1.0	20	0.987
19	100	0.01	0.5	2.0	60	0.987

Table 4.2.1: Optimal parameters for graph 000 with first-order bit-flip constraints ($N_X = 5$ for all trials). All trials are run for $T = 150$ epochs. We tabulate parameters that yield the best performance, with cut ratios of $C_Q/C_{SDP} = 0.987$.

Graph	c_b	η	α	r	d	$\overline{C_Q}$	C_{SDP}	$\overline{C_Q}/C_{\text{SDP}}$
001	100	0.005	0.5	1.0	40	223.67	230	0.972
002	100	0.01	0.01	1.5	40	224.33	231	0.971
003	150	0.005	0.05	2.0	60	228.33	236	0.967
004	100	0.005	1.00	2.0	60	220.67	229	0.963
005	150	0.005	0.5	1.0	40	228.33	236	0.967
006	100	0.01	1.0	1.0	60	229.00	233	0.983
007	100	0.01	0.05	3.0	20	225.33	234	0.963
008	100	0.005	0.01	2.0	60	218.67	225	0.971
009	150	0.005	0.5	1.0	60	235.00	240	0.979

Table 4.2.2: Optimal parameters for graphs 001 through 009 with first-order bit-flip constraints, $N_X = 5$ for all graphs. We show the best performing parameters for each graph for three repetitions, and report the average HTAAC-QSDP cut $\overline{C_Q}$, C_{SDP}, and the average cut ratio $\overline{C_Q}/C_{\text{SDP}}$. All trials are run for $T = 150$ epochs.

Chapter 5

Conclusion and future work

In this thesis, we experimentally show that the Hadamard test with approximate amplitude constraints (HTAAC-QSDP) extends to MaxBisection, with solution quality equivalent to the best classical SDP methods. The method, as introduced by Patti et al., [31] uses $n + 1$ qubits to solve SDPs with $N = 2^n$ variables and $M \sim 2^n$ constraints by taking a constant number of quantum measurements (two Hadamard tests for U_W, U_P) and $\mathcal{O}(n^2)$ classical calculations per epoch. More specifically, we estimate $\mathcal{O}(n^2)$ approximate amplitude constraints and $\mathcal{O}(n)$ first-order bit-flip constraints for MaxBisection. For this problem, this represents an exponential reduction in the number of required expectation values.

We use a quantum implementation of Goemans-Williamson to approximately enforce the $M = 2^n$ constraints using a population balancing Hadamard test. We demonstrate this method on ten generated Erdös-Rényi graphs, approaching the performance of leading gradient-based classical SDP solvers on all graphs.

In future work, these experiments can be tested on additional types of graphs like toroid graphs or skewed binary and skewed integer graphs, as is done in by Patti et al. [31]. Moreover, this method can be extended to additional problems like Max2Sat or to additional types of SDPs.

Appendix A

Additional plots and figures

Here, we present plots and figures that supplement results and analysis.

We have the average loss and average cut ratio for graphs 001 through 004 shown in Fig. A.0.1. We generate plots using $\eta = 0.005$, $c_b = 100$, $r = 1.2$, $T = 150$, $d = 120$ and 10 repetitions per trial, to obtain average values.

Moreover, we tabulate the five highest-performing sets of parameters for graphs 001 through 009 in Tab. A.0.1., using optimal parameters obtained from two rounds of hyperparameter searches on graph 000. We observe that these parameters yield a cut ratio of at least 0.95 for all graphs, suggesting that HTAAC-QSDP generalizes well to graphs outside of the training set.

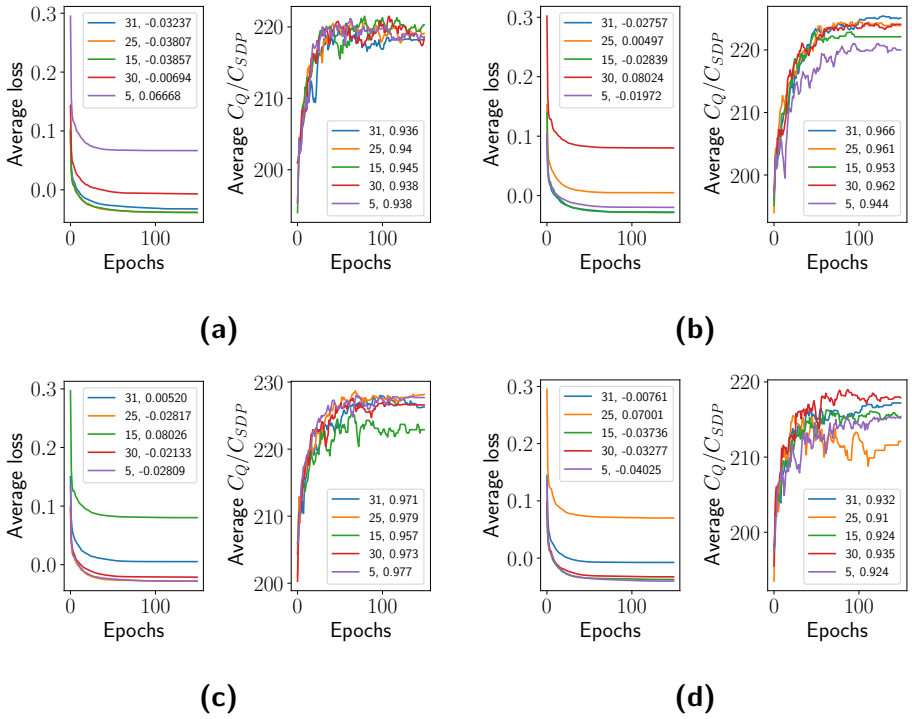


Figure A.0.1: Plots generated using a learning rate $\eta = 0.005$, coefficient base $c_b = 100$, U_P regularization $r = 1.2$, $T = 150$, a circuit d of 120 and 10 repetitions per trial. Each pair contains the average loss and cut ratio, respectively, of 10 repetitions after 150 epochs using different number of Pauli-x string constraints. The legend is in the form (c, v), where c is the number of constraints and v is the (rounded) value at time T. A.0.1a Plots for graph 001. A.0.1b Plots for graph 002. A.0.1c Plots for graph 003. A.0.1d Plots for graph 004.

Graph	c_b	η	a	r	d	$\overline{C_Q}/C_{SDP}$	Graph	c_b	η	a	r	d	$\overline{C_Q}/C_{SDP}$
001	100	0.005	0.5	1.0	40	0.972	002	100	0.01	0.01	1.5	40	0.971
001	100	0.01	1.0	1.0	40	0.972	002	100	0.01	0.5	2.0	60	0.967
001	150	0.005	0.5	1.0	60	0.970	002	100	0.005	0.1	2.0	60	0.960
001	100	0.01	0.01	1.5	40	0.970	002	100	0.01	0.05	1.0	20	0.958
001	100	0.005	1.0	2.0	60	0.968	002	100	0.005	0.5	1.0	40	0.958
003	100	0.005	0.05	2.0	60	0.968	004	100	0.005	1.0	2.0	60	0.964
003	100	0.01	1.0	1.0	60	0.965	004	200	0.01	0.01	2.0	60	0.962
003	100	0.0025	0.5	1.5	60	0.962	004	100	0.01	1.0	1.0	60	0.961
003	150	0.01	0.05	1.0	40	0.962	004	100	0.0025	1.0	1.0	40	0.961
003	100	0.005	0.01	2.0	60	0.960	004	150	0.005	0.5	1.0	60	0.951
005	100	0.005	0.5	1.0	40	0.968	006	100	0.01	1.0	1.0	60	0.983
005	100	0.0025	1.0	1.0	40	0.968	006	100	0.005	0.01	2.0	60	0.976
005	150	0.005	0.5	1.0	60	0.966	006	100	0.005	1.0	2.0	60	0.973
005	100	0.01	1.0	1.0	40	0.962	006	100	0.01	1.0	1.0	40	0.970
005	200	0.01	0.01	1.0	60	0.960	006	100	0.0025	0.5	1.5	60	0.969
007	100	0.01	0.05	1.0	20	0.963	008	100	0.005	0.01	2.0	60	0.972
007	150	0.005	0.5	1.0	60	0.962	008	100	0.005	0.05	2.0	60	0.967
007	100	0.005	0.01	1.0	20	0.960	008	100	0.005	0.05	1.5	60	0.967
007	100	0.01	1.0	1.0	60	0.960	008	100	0.0025	0.5	1.5	60	0.963
007	100	0.005	0.01	2.0	60	0.956	008	100	0.005	0.1	2.0	60	0.960
009	150	0.005	0.5	1.0	60	0.979							
009	100	0.005	0.01	2.0	60	0.975							
009	100	0.01	0.01	1.5	40	0.971							
009	100	0.005	1.0	2.0	60	0.971							
009	100	0.005	0.05	2.0	60	0.969							

Table A.0.1: Optimal parameters for graphs 001 through 009 with first-order bit-flip constraints ($N_X = 5$ for all trials). All trials are run for $T = 150$ epochs. Recall we have the coefficient base c_b, learning rate η, unitary phase a, population regularization r, circuit depth d, mean HTAAC-QSDP cut $\overline{C_Q}$, and mean performance or cut ratio $\overline{C_Q}/C_{SDP}$. We observe that there is not much variance in the performance over best parameters.

References

[1] Richard P Feynman. Simulating physics with computers. In *Feynman and computation*, pages 133–153. CRC Press, 2018.

[2] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings 35th annual symposium on foundations of computer science*, pages 124–134. Ieee, 1994.

[3] Aram W Harrow, Avinatan Hassidim, and Seth Lloyd. Quantum algorithm for linear systems of equations. *Physical review letters*, 103(15):150502, 2009.

[4] Seth Lloyd. Universal quantum simulators. *Science*, 273(5278):1073–1078, 1996.

[5] Yudong Cao, Jhonathan Romero, and Alán Aspuru-Guzik. Potential of quantum computing for drug discovery. *IBM Journal of Research and Development*, 62(6):6–1, 2018.

[6] Román Orús, Samuel Mugel, and Enrique Lizaso. Quantum computing for finance: Overview and prospects. *Reviews in Physics*, 4:100028, 2019.

[7] John Preskill. Quantum computing and the entanglement frontier. *arXiv preprint arXiv:1203.5813*, 2012.

[8] Frank Arute, Kunal Arya, Ryan Babbush, Dave Bacon, Joseph C Bardin, Rami Barends, Rupak Biswas, Sergio Boixo, Fernando GSL Brandao, David A Buell, et al. Quantum supremacy using a programmable superconducting processor. *Nature*, 574(7779):505–510, 2019.

[9] Han-Sen Zhong, Hui Wang, Yu-Hao Deng, Ming-Cheng Chen, Li-Chao Peng, Yi-Han Luo, Jian Qin, Dian Wu, Xing Ding, Yi Hu, et al. Quantum

computational advantage using photons. *Science*, 370(6523):1460–1463, 2020.

[10] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cincio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.

[11] Edward Farhi, Jeffrey Goldstone, Sam Gutmann, and Michael Sipser. Quantum computation by adiabatic evolution. *arXiv preprint quant-ph/0001106*, 2000.

[12] Tameem Albash and Daniel A Lidar. Adiabatic quantum computation. *Reviews of Modern Physics*, 90(1):015002, 2018.

[13] Sepehr Ebadi, Alexander Keesling, Madelyn Cain, Tout T Wang, Harry Levine, Dolev Bluvstein, Giulia Semeghini, Ahmed Omran, J-G Liu, Rhine Samajdar, et al. Quantum optimization of maximum independent set using rydberg atom arrays. *Science*, page eabo6587, 2022.

[14] Tadashi Kadowaki and Hidetoshi Nishimori. Quantum annealing in the transverse ising model. *Physical Review E*, 58(5):5355, 1998.

[15] Elizabeth Gibney. D-wave upgrade: How scientists are using the world's most controversial quantum computer. *Nature*, 541(7638), 2017.

[16] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[17] Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[18] Lieven Vandenberghe and Stephen Boyd. Applications of semidefinite programming. *Applied Numerical Mathematics*, 29(3):283–299, 1999.

[19] Wenjun Li, Yang Ding, Yongjie Yang, R Simon Sherratt, Jong Hyuk Park, and Jin Wang. Parameterized algorithms of fundamental np-hard problems: A survey. *Human-centric Computing and Information Sciences*, 10(1):1–24, 2020.

[20] Michel X Goemans and David P Williamson. .879-approximation algorithms for max cut and max 2sat. In *Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 422–431, 1994.

[21] Michel X Goemans. Semidefinite programming in combinatorial optimization. *Mathematical Programming*, 79(1):143–161, 1997.

[22] David P Williamson and David B Shmoys. *The design of approximation algorithms*. Cambridge university press, 2011.

[23] Fernando GSL Brandão, Amir Kalev, Tongyang Li, Cedric Yen-Yu Lin, Krysta M Svore, and Xiaodi Wu. Quantum sdp solvers: Large speed-ups, optimality, and applications to quantum learning. *arXiv preprint arXiv:1710.02581*, 2017.

[24] Joran van Apeldoorn and András Gilyén. Improvements in quantum sdp-solving with applications. *arXiv preprint arXiv:1804.05058*, 2018.

[25] Joran Van Apeldoorn, András Gilyén, Sander Gribling, and Ronald de Wolf. Quantum sdp-solvers: Better upper and lower bounds. *Quantum*, 4:230, 2020.

[26] Fernando GSL Brandao and Krysta M Svore. Quantum speed-ups for solving semidefinite programs. In *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 415–426. IEEE, 2017.

[27] Fernando GS L Brandao, Richard Kueng, and Daniel Stilck França. Faster quantum and classical sdp approximations for quadratic binary optimization. *Quantum*, 6:625, 2022.

[28] Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of computing*, 8(1):121–164, 2012.

[29] Dhrumil Patel, Patrick J Coles, and Mark M Wilde. Variational quantum algorithms for semidefinite programming. *arXiv preprint arXiv:2112.08859*, 2021.

[30] Kishor Bharti, Tobias Haug, Vlatko Vedral, and Leong-Chuan Kwek. Nisq algorithm for semidefinite programming. *arXiv preprint arXiv:2106.03891*, 2021.

[31] Taylor L Patti, Jean Kossaifi, Anima Anandkumar, and Susanne F Yelin. Quantum semidefinite programming with the hadamard test and approximate amplitude constraints. *arXiv preprint arXiv:2206.14999*, 2022.

[32] Michel X Goemans and David P Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM (JACM)*, 42(6):1115–1145, 1995.

[33] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[34] Robert L Kosut and Daniel A Lidar. Quantum error correction via convex optimization. *Quantum Information Processing*, 8:443–459, 2009.

[35] Yi-Kai Liu, Matthias Christandl, and Frank Verstraete. Quantum computational complexity of the n-representability problem: Qma complete. *Physical review letters*, 98(11):110503, 2007.

[36] Christoph Helmberg. *Semidefinite programming for combinatorial optimization*. PhD thesis, 2000.

[37] László Lovász and Alexander Schrijver. Cones of matrices and set-functions and 0–1 optimization. *SIAM journal on optimization*, 1(2):166–190, 1991.

[38] Etienne de Klerk, Cornelis Roos, and Tamás Terlaky. *Semi-definite problems in truss topology optimization*. Delft University of Technology, Faculty of Technical Mathematics and Informatics, 1995.

[39] Pablo A Parrilo. Semidefinite programming relaxations for semialgebraic problems. *Mathematical programming*, 96:293–320, 2003.

[40] Horace Yuen, Robert Kennedy, and Melvin Lax. Optimum testing of multiple hypotheses in quantum detection theory. *IEEE transactions on information theory*, 21(2):125–134, 1975.

[41] Yonina C Eldar. A semidefinite programming approach to optimal unambiguous discrimination of quantum states. *IEEE Transactions on information theory*, 49(2):446–456, 2003.

[42] Xin Wang, Wei Xie, and Runyao Duan. Semidefinite programming strong converse bounds for classical capacity. *IEEE Transactions on Information Theory*, 64(1):640–653, 2017.

[43] Xin Wang. *Semidefinite optimization for quantum information*. PhD thesis, 2018.

[44] Ivan Šupić and Joseph Bowles. Self-testing of quantum systems: a review. *Quantum*, 4:337, 2020.

[45] Florian A Potra and Stephen J Wright. Interior-point methods. *Journal of computational and applied mathematics*, 124(1-2):281–302, 2000.

[46] Sanjeev Arora and Satyen Kale. A combinatorial, primal-dual approach to semidefinite programs. *Journal of the ACM (JACM)*, 63(2):1–35, 2016.

[47] Alan Frieze and Mark Jerrum. Improved approximation algorithms for max k-cut and max bisection. *Algorithmica*, 18(1):67–81, 1997.

[48] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[49] Michael R Garey, David S Johnson, and Larry Stockmeyer. Some simplified np-complete problems. In *Proceedings of the sixth annual ACM symposium on Theory of computing*, pages 47–63, 1974.

[50] Dimitri P Bertsekas. *Constrained optimization and Lagrange multiplier methods*. Academic press, 2014.

[51] Alan J Soper, Chris Walshaw, and Mark Cross. A combined evolutionary search and multilevel optimisation approach to graph-partitioning. *Journal of Global Optimization*, 29(2):225–241, 2004.

[52] Aric Hagberg, Pieter Swart, and Daniel S Chult. Exploring network structure, dynamics, and function using networkx. Technical report, Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[53] Erdős Paul and Rényi Alfréd. On random graphs i. *Publicationes Mathematicae (Debrecen)*, 6:290–297, 1959.

[54] Béla Bollobás and Béla Bollobás. *Random graphs*. Springer, 1998.

[55] Steven Diamond and Stephen Boyd. CVXPY: A Python-embedded modeling language for convex optimization. *Journal of Machine Learning Research*, 17(83):1–5, 2016.

[56] Akshay Agrawal, Robin Verschueren, Steven Diamond, and Stephen Boyd. A rewriting system for convex optimization problems. *Journal of Control and Decision*, 5(1):42–60, 2018.

[57] Taylor L Patti, Jean Kossaifi, Susanne F Yelin, and Anima Anandkumar. Tensorly-quantum: Quantum machine learning with tensor methods. *arXiv preprint arXiv:2112.10239*, 2021.

[58] Jean Kossaifi, Yannis Panagakis, Anima Anandkumar, and Maja Pantic. Tensorly: Tensor learning in python. *arXiv preprint arXiv:1610.09555*, 2016.

[59] Charles R Harris, K Jarrod Millman, Stéfan J Van Der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J Smith, et al. Array programming with numpy. *Nature*, 585(7825):357–362, 2020.

[60] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.