Performance of Quantum SDP Methods for Quadratic Binary Optimization Problems

Dissertation zur Erlangung des Doktorgrades
in Theoretischer Physik

der
Mathematisch-Naturwissenschaftlichen Fakultät
der Universität zu Köln

vorgelegt von

Fabian Henze

Köln, 2025

Diese Dissertation wurde von der Mathematisch-Naturwissenschaftlichen Fakultät der Universität zu Köln angenommen.

Gutachter: Prof. Dr. David Gross
Prof. Dr. Felix Motzoi

Tag der Disputation: 24.06.2025

Abstract

Quantum computing algorithms offer asymptotic speedups compared to classical approaches in many domains. However, it remains uncertain whether these advantages extend to problem instances that can be solved within a reasonable time. One challenge for quantum algorithms is their often high scaling cost with respect to precision. Consequently, applications that allow approximate solutions are more promising candidates for quantum speedups. One such example are *semidefinite programming* (SDP) relaxations of *quadratic binary optimization* problems, which incorporate a rounding procedure at the end. The *Hamiltonian Updates* (HU) SDP-solving algorithm introduced by Brandão et al. in 2019 was specifically designed for such problems. This thesis analyzes the performance of the HU algorithm and assesses whether it could provide a quantum advantage. To ensure a fair comparison, we develop heuristics that significantly improve the algorithm's efficiency. Since large-scale quantum computers do not yet exist, direct benchmarking is not possible. Instead, we adopt a *hybrid* approach, estimating the running time of quantum subroutines by analyzing their required number of gate operations. Our findings suggest that even with these optimizations and under highly optimistic assumptions favoring the quantum approach, a quantum implementation of the Hamiltonian Updates algorithm is unlikely to outperform classical methods for realistic problem sizes. Moreover, when applying SDP relaxations to real-world datasets, we observe that approximate relaxation techniques can lead to highly suboptimal solutions for the original optimization problems.

Contents

Chapter 1

Introduction

In recent years, quantum computing has gained an enormous amount of public attention. Contributing to this are several announcements of achieving a *quantum advantage* (i.e. solving a problem faster on a quantum computer than it could be done on a classical computer) for specific problems like in the "quantum computational supremacy" experiment by the Google AI Quantum group in 2019 [Aru+19], and others [Zho+21; Drm+25]. However, one has to note that these experiments only serve as *proof-of-concepts*, as the selected problems are specifically designed to demonstrate quantum advantage and do not have real-world applications [SK22]. Furthermore, it is crucial to compare quantum performance against the best-known classical algorithms. For Google's claim this was not the case, as IBM pointed out soon after the announcement [Res19].

The gap to developing a quantum device capable of outperforming classical computers on practical problems remains substantial. Nevertheless, significant advances in the scale and fidelity of quantum computers [Phi+22; Xue+22; PKG24] spark hope that quantum computers could show advantages on practical problems in the not too far future. A particular group of possible applications that arise for quantum computing are optimization problems. There are countless quantum algorithms presented in academic literature designed for optimization that show better asymptotic scaling than classical algorithms. But this does not necessarily lead to an actual advantage of quantum implementation for practical problems. Indeed, several studies have shown that, for specific algorithms, quantum computers are not expected to achieve a practical advantage [Dal+23; Amm+23].

A frequently considered class of optimization problems in quantum computing are *quadratic unconstrained binary optimization* (QUBO) problems. In QUBOs, the complete optimization task is encoded within a single cost matrix, without any further constraints. The compact structure of QUBOs makes them a popular choice for formulating problems in quantum algorithms, as they enable more general and simpler routines. However, QUBO problems are known to be NP-hard in general, and it is widely believed that this class of problems cannot be solved efficiently (i.e. in polynomial time with respect to problem size) even on a quantum computer [Aar05; Ben+97]. Therefore, it is also of interest to find efficient ways to approximate QUBOs. This can be done by relaxations of the problem as *semidefinite programs* (SDPs). Two examples of methods that build on such SDP relaxations are the

Goemans-Williamson (GW) algorithm and *sum-of-squares* (SOS) method. We will study these two in this work.

Although, the SDPs arising in the GW algorithm can be solved classically in polynomial time, this scaling can still quickly become too costly for larger problem instances. In 2019, Ref. [GLBKSF22] introduced an algorithm inspired by quantum mechanics called *Hamiltonian Updates* (HU) that allows for certain subroutines to be implemented on a quantum computer with favorable asymptotic scaling in the problem dimension. The main goal of this thesis is to estimate the performance of the quantum implementation of HU and assess whether a quantum advantage on potential future quantum hardware is possible. For this, we first focus on significantly improving the practical performance of the HU algorithm. This is required for a fair comparison, as the original publication mainly focused on optimizing the asymptotic performance. In a second step, we classically benchmark the HU implementation, while simultaneously tracking and bounding the minimum number of required gate operations for a quantum implementation.

The structure of the thesis is as follows: In Chapter 2 we give an overview of the optimization problems discussed in this thesis. Chapter 3 discusses hurdles for quantum algorithms to provide a practical advantage, as well as obstacles and approaches for quantifying their performance. Chapter 4 summarizes and includes the publication [Hen+25] that forms the main part of this work, where the HU algorithm is improved and benchmarked. Building on this, Chapter 5 uses the techniques introduced in [Hen+25] to benchmark HU for problems with real-world data and compare these to other techniques including the SOS method. Finally, in Chapter 6 we summarize the results and discuss their relevance for quantum based optimization algorithms in general.

Chapter 2

Optimization

2.1 Basics of optimization problems

Optimization problems are often referred to as *programs*, and the process of solving them is known as *programming*. In an optimization problem, the goal is to find the optimal value of a variable x, which is restricted to a domain A called the *search space*.[1] The objective is to maximize or minimize a given *cost function* or *objective function* $f : A \to \mathbb{R}$, which assigns an *objective value* to each x. Additionally, x is often subject to a set of constraints. Finding the exact optimal solution is often challenging. As a result, it is useful to develop efficient methods for computing bounds on the optimal value.

Optimization problems can be broadly classified into two categories based on the nature of the search space: (i) **discrete optimization**, where A is a countable set, meaning x takes discrete values (e.g. integers or binary values) and (ii) **continuous optimization**, where A is a continuous set, allowing x to take real values.

Continuous optimization problems are usually easier to solve than discrete ones. A common approach to discrete problems is to *relax* them by extending their search space to a continuous domain, for example, allowing x to take real values instead of integers. Since the original search space is a subset of the relaxed one, the optimal objective value of the relaxed problem serves as an *upper bound* for a maximization problem.

Relationship between optimization and feasibility problems

A common approach in optimization is to reformulate an optimization problem

$$\underset{x \in A}{\text{maximize}} \qquad f(x) \tag{2.1}$$

as a *feasibility problem*

$$\begin{aligned} \text{find} \quad & x \in A \\ \text{subject to} \quad & f(x) \leq \gamma, \end{aligned} \tag{2.2}$$

[1] Typically, x is a vector, but depending on the problem structure, it may be more convenient to represent it as a matrix. In such cases, the capitalized notation X is commonly used.

where the feasibility formulation introduces an additional parameter $\gamma \in \mathbb{R}$ as a threshold for the objective value. If we can determine whether Equation (2.2) is *feasible* (i.e. there exists a value for x that fulfills the constraints) for a given γ, we can approximate the optimal objective value using binary search. The overhead introduced by binary search scales only logarithmically with the target precision, i.e. the gap between the largest infeasible and smallest feasible value of γ.

NP-hardness

The complexity class *nondeterministic polynomial time* (NP) includes all decision problems for which a given solution (e.g. a value for x) can be verified in polynomial time. However, this does not imply that such a solution can be found efficiently. The hardest problems within NP are known as *NP-complete*. Any problem in NP, including other NP-complete problems, can be *reduced* to an NP-complete problem with only a polynomial number of calls to that problem. Problems that are at least as hard as NP-complete problems are called *NP-hard*. It is widely believed, though not proven, that NP-hard problems cannot be solved in polynomial time [GJ79].

2.2 Integer programming

A wide range of industry problems can be expressed as *integer programs* (IPs), ranging from scheduling problems and electricity generation planning to optimization of kidney exchange programs [Wol20]. IPs are characterized by a search space restricted to integer values. If a problem involves both integer and continuous variables, it is referred to as a *mixed integer program* (MIP). In general, solving IPs is NP-hard. However, significant progress has been made in recent years with modern solvers capable of handling large-scale problems efficiently [CL24]. These include commercial solvers such as *Gurobi* [Gur24] and *CPLEX* [Cpl09], as well as open-source alternatives like *SCIP* [Bol+24].

IPs can be categorized based on the structure of their objective function. Below, we introduce two important classes of IPs, which will be used later for benchmarking with real-world instances in Chapter 5.

Integer linear programs

An *integer linear program* (ILP) takes the form

$$\begin{aligned} \underset{x \in \mathbb{Z}^n,\ x \geq 0}{\text{maximize}} \quad & c^T x \\ \text{subject to} \quad & Ax \leq b, \end{aligned} \tag{2.3}$$

where $c \in \mathbb{R}^n$ is the *cost vector*, and $A \in \mathbb{R}^{k \times n}, b \in \mathbb{R}^k$ define k linear constraints on x.

Integer quadratic programs

A generalization of ILPs is the *integer quadratic program* (IQP), which includes quadratic terms in the objective function:

$$\begin{aligned} \underset{x\in\mathbb{Z}^n,\, x\geq 0}{\text{maximize}} \quad & x^T C x + c^T x \\ \text{subject to} \quad & Ax \leq b, \end{aligned} \tag{2.4}$$

where $C \in \mathbb{R}^{n\times n}$ is a quadratic cost matrix.

2.3 Quadratic unconstrained binary optimization

Another common formulation of optimization problems is *quadratic unconstrained binary optimization* (QUBO). In a QUBO problem, the goal is to optimize a quadratic objective function over a binary vector x. It is formulated as

$$\underset{x\in\{-1,1\}^n}{\text{maximize}} \quad x^T C x, \tag{2.5}$$

where $C \in \mathbb{R}^{n\times n}$ is called the *cost matrix*. An alternative formulation restricts x to $\{0,1\}^n$ instead of $\{-1,1\}^n$. These representations are mathematically equivalent and can be converted into one anothe (see e.g. [BJR89]).[2]

Similar to IPs, QUBO problems are NP-hard in general. IPs can be transformed into QUBOs by incorporating constraints as penalty terms in the objective function (see Section 5.2). Due to their unconstrained nature, QUBOs are often preferred as a starting point in theoretical optimization literature [Luc14], particularly in quantum computing, where various quantum algorithms, such as variational methods [McC+16] and quantum annealing [Far+01], are designed to solve them.

We now give two examples of optimization problems that can be formulated as QUBOs.

MaxCut

MaxCut is one of Karp's famous NP-complete problems [Kar72]. It is defined on an undirected graph $G = (V, E)$ with edge weights $w : E \to \mathbb{R}$. The goal is to partition the set of vertices V into two groups such that the sum of the weights of the edge $E_{\text{cut}} \subset E$ connecting the two groups is maximized:

$$\underset{E_{\text{cut}}\subset E}{\text{maximize}} \sum_{e\in E_{\text{cut}}} w(e). \tag{2.6}$$

By introducing a binary vector $x \in \{-1,1\}^n$ with $n = |V|$ to represent the partitioning of the vertices, and defining the adjacency matrix W, the objective function

[2] When converting to the $\{-1,1\}$ representation the dimension increases by one.

can be rewritten as

$$\sum_{e \in E_{\text{cut}}} w(e) = \sum_{e \in E} w(e) - \frac{1}{2} x^T W x. \tag{2.7}$$

Since the constant term $\sum_{e \in E} w(e)$ does not affect the optimization, the MaxCut problem (2.6) can be formulated as a QUBO (2.5) with $C = -W/2$.

Cut norm

The *cut norm* of a matrix $B \in \mathbb{R}^{m \times m}$ is defined as the optimal value of the following optimization problem:

$$\underset{y,z \in \{-1,1\}^m}{\text{maximize}} \qquad y^T B z. \tag{2.8}$$

This problem can be expressed as a QUBO. To do so, we define the block matrix

$$C = \frac{1}{2} \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}. \tag{2.9}$$

Next, we introduce the binary vector $x \in \mathbb{R}^n$ with $n = 2m$, constructed as the direct sum of y and z, i.e. $x = y \oplus z$. This allows us to rewrite the objective function as $y^T B z = x^T C x$. Thus, the cut norm of B is equal to the optimal value of the QUBO formulation

$$\underset{x \in \{-1,1\}^n}{\text{maximize}} \qquad x^T C x. \tag{2.10}$$

2.4 Semidefinite programming

A fundamental class of matrices is the set of *(semi-)definite matrices*. A matrix $A \in \mathbb{R}^{n \times n}$ is said to be *positive semidefinite* (psd) if

$$x^T A x \geq 0 \qquad \text{for all } x \in \mathbb{R}^n.$$

This is denoted by $X \succeq 0$. If the inequality in (2.11) is strict for all nonzero x, then A is *positive definite*. If the inequality is reversed, A is *negative (semi-)definite*. A symmetric matrix is psd if and only if all its eigenvalues are nonnegative.

Positive semidefinite matrices play an essential role in various fields, including optimization and quantum mechanics. For instance, density matrices representing quantum states are psd. Another application is in optimization, where psd matrices are part of *semidefinite programming* (SDP) [Tod01].

Semidefinite programming formulation

Semidefinite programming refers to a class of convex optimization problems where the optimization variable is a psd matrix. A standard SDP formulation is given by

$$\begin{aligned} \underset{X \in \mathbb{R}^{n \times n}}{\text{maximize}} \quad & \operatorname{tr}(CX) \\ \text{subject to} \quad & \operatorname{tr}(A_j X) \leq b_j, \\ & X \succeq 0. \end{aligned} \tag{2.11}$$

SDPs can be efficiently solved using interior-point methods [NN94; Ali95], making them a valuable tool in combinatorial optimization, control theory [BV94], and machine learning [Lan+04]. We will discuss an application of SDPs for relaxing QUBOs in the next section.

2.5 Goemans-Williamson algorithm

The *Goemans-Williamson algorithm* [GW95] is a well-known approximation algorithm that relaxes a QUBO problem into an SDP. The idea is to lift the quadratic binary optimization problem to a continues higher-dimensional space where it becomes a convex problem.

SDP Relaxation of QUBO

The objective function of a QUBO problem is given by $x^T C x$ with $x \in \{-1, 1\}$. This can be rewritten using the outer product as

$$x^T C x = \operatorname{tr}(C x x^T) = \operatorname{tr}(CX),$$

where $X \in \mathbb{R}^{n \times n}$ is defined as $X = xx^T$. Now, X is a psd, rank-1 matrix with all diagonal entries equal to 1. We can relax the problem by removing the rank constraint while preserving positive semidefiniteness and diagonal constraints. The resulting SDP relaxation is

$$\begin{aligned} \underset{X \in \mathbb{R}^{n \times n}}{\text{maximize}} \quad & \operatorname{tr}\left(C^T X\right) \\ \text{subject to} \quad & \operatorname{diag}(X) = 1, \\ & X \succeq 0. \end{aligned} \tag{2.12}$$

While the original QUBO is NP-hard, the SDP relaxation can be solved in polynomial time.

Randomized rounding

To obtain a binary vector $x \in \{-1, 1\}$ from the relaxed solution X, Goemans and Williamson proposed a *randomized rounding* technique. This procedure involves:

1. Computing the square root decomposition of X, i.e. finding $\sqrt{X}$ such that $\sqrt{X}\sqrt{X}^T = X$.
2. Sampling a random Gaussian vector $g \sim \mathcal{N}(0,1)$, $g \in \mathbb{R}^n$.
3. Projecting $\sqrt{X}$ column-wise onto g and taking the sign to obtain a binary vector:

$$x_i = \operatorname{sign}\left(\sum_{j\in[n]} (\sqrt{X})_{ij} g_j\right), \quad i \in [n].$$

Since solving the SDP and computing $\sqrt{X}$ is typically more expensive than the rounding procedure, it is often beneficial to repeat the rounding step multiple times and select the best result.

Approximation guarantees

For certain cost matrices C, the expected objective value of the rounded solution can be bounded by a constant factor $\alpha \in [0,1]$ of the optimal SDP value. Let $X^\star \in \mathbb{R}^{n\times n}$ be the optimal SDP solution and $x^\star \in \{-1,1\}^n$ the optimal QUBO solution. Define $x^{(g)} \in \{-1,1\}^n$ as the binary vector obtained by applying the randomized rounding procedure to $X^\star$ with the random Gaussian vector g.[3] Then, for certain cost matrices the expected objective value after rounding taken over the distribution of Gaussian vectors g can be lower bounded as follows:

$$\mathbb{E}_g[(x^{(g)})^T C x^{(g)}] \geq \alpha \operatorname{tr}(CX^\star) \geq \alpha (x^\star)^T C x^\star. \tag{2.13}$$

The second inequality follows from the fact that the SDP is a relaxation of the original QUBO. The first inequality requires more involved proofs, and the value of α depends on the structure of C. Some known bounds include:

- If the entries of C are nonpositive, we have $\alpha \approx 0.878$. If the *Unique games conjecture* holds, this bound is optimal [Kho+07].
- If C is psd, [AN06] prove a bound of $\alpha = \frac{2}{\pi}$
- If C takes the block form of Equation (2.9) corresponding to the cut norm problem, [AN06] prove a bound o $\alpha = \frac{\pi}{4} - 1$

Note that the bound in (2.13) may not always be meaningful, e.g. if $\operatorname{tr}(CX^\star)$ is negative (e.g. for $C = -\mathbb{1}$). The guarantees listed above typically assume that the SDP relaxation is solved exactly. In Section 4.2, we derive adjusted bounds for inexact solutions in the cut norm case.

[3] In the publication included in Section 4.2 we usually omit to denote the dependence on g.

Chapter 3

Performance of quantum algorithms

The potential computational advantage of quantum computers over classical ones stems from their ability to store and process information in a superposition of states. Simulating such a system on a classical computer would require exponentially more memory and computational operations. One might therefore think that quantum computing is equivalent to performing an exponentially large number of computations in parallel. However, this is not the case, as it is not possible to extract all the information contained in a quantum superposition. More precisely, the maximum amount of classical bits of information that can be retrieved from measurements on a quantum computer is limited to the number of qubits. Moreover, since a complete measurement causes a collapse of the quantum superposition, additional information cannot be obtained through repeated measurements without rerunning the quantum computation.

To achieve a computational advantage, quantum algorithms must manipulate the quantum state in such a way that measurements yield exactly the desired information. The process for this depends on the specific problem being solved. In some cases, this can be done efficiently, enabling quantum computers to achieve exponential (or at least superpolynomial) speedups. A prominent example is *Shor's factoring algorithm* [Sho97]. For other problems, significantly more computational overhead is required to extract useful information from the quantum state. A typical example is *Grover's search algorithm* [Gro96], which amplifies the probability of measuring the correct solution by gradually increasing the amplitude of the target state in a sequence of rotations and reflections. This technique results only in a quadratic speedup, rather than an exponential one. Problems of this nature are usually found in combinatorial optimization. Therefore, these generally do not exhibit the same dramatic quantum advantage as problems like integer factorization.

3.1 NISQ vs. fault-tolerant algorithms

When discussing quantum algorithm, one typically distinguishes between two categories: (i) algorithms designed for *noisy intermediate-scale quantum* (NISQ) hardware, which lacks error correction capabilities, and (ii) algorithms that require *fault-tolerant* quantum computing [Got98], where error correction ensures reliable computations. While NISQ hardware is expected to become available in the near future [Pre18], fully fault-tolerant quantum computers remain a distant goal [Bev+22].

Fault tolerance imposes strict requirements on quantum gate fidelities and incurs significant overhead in terms of the number of physical qubits and gate operations required for error correction.

NISQ algorithms

NISQ algorithms typically include *variational algorithms*, such as the *variational quantum eigensolver* (VQE) [Per+14] and the *quantum approximate optimization algorithm* (QAOA) [FGG14]. These algorithms rely on relatively short quantum circuits with tunable parameters. By repeatedly executing the quantum routine and optimizing these parameters based on the measurement outcomes, one aims to find an optimal configuration that yields a good approximation to the original problem. Due to the complex structure arising from these quantum operations, the performance of variational algorithms is often difficult to predict theoretically. Some theoretical results provide insights into their solution quality, including both positive results (e.g. [Far+22; BM24]) and negative results (e.g. [Bra+20; DP+23; FGG20]). However, while these results are impressive from a mathematical standpoint, they often apply only to highly structured or even trivial problem instances. It remains an open question whether variational algorithms can outperform classical algorithms on practically relevant problems.

Faul-tolerant algorithms

Fault-tolerant quantum algorithms, in contrast, rely on deeper quantum circuits, making them more susceptible to errors. To ensure reliability, they require high-fidelity quantum gates and error correction protocols. The latter typically involve encoding each logical qubit in multiple physical qubits, leading to substantial resource overheads. Unlike NISQ algorithms, fault-tolerant quantum algorithms provide rigorous theoretical guarantees, ensuring they return the correct result with high probability. This makes them more predictable in terms of performance but also significantly more challenging to implement on hardware.

3.2 Asymptotic vs. non-asymptotic performance

In scientific literature, algorithms are often evaluated based on their asymptotic performance, meaning their running time or resource consumption is expressed in terms of how they scale with input size. An algorithm with better asymptotic complexity will eventually outperform another for sufficiently large instances. However, asymptotic improvements do not indicate where the crossover point lies. In many cases, an algorithm may theoretically surpass another but only at input sizes so large that they are infeasible to implement on any existing hardware. This class of algorithms is sometimes referred to as "galactic algorithms".

Another important distinction is the difference between worst-case and average-case performance. For example, while the *simplex algorithm* for linear programming has been shown to require exponential time in worst-case instances [KM72], it often performs extremely well in practice, making it a preferred choice over *interior point*

methods, which are proven to run in polynomial time [Kar84]. Beyond time complexity, numerical stability is another relevant factor. For instance, *Strassen's algorithm* for matrix multiplication [Str69] has an improved asymptotic complexity of $O(n^{2.81})$ compared to the standard $O(n^3)$ method. However, due to stability issues [DSGW18], it is rarely used in practice.

3.3 Non-asymptotic performance of quantum algorithms

Quantum algorithms often suffer from worse non-asymptotic performance compared to their classical counterparts. Several factors contribute to this disadvantage:

- **Slower gate operations:** Quantum gates are significantly more challenging to implement physically than classical gates. As a result, even with optimistic projections for future quantum hardware, the average execution time for a quantum gate operation is expected to be substantially slower than for a classical operation.
- **Overhead from error correction:** The necessity of quantum error correction introduces a significant resource overhead, increasing both the number of required physical qubits and the overall execution time.
- **Stochastic nature of quantum algorithms:** Unlike classical deterministic computations, quantum algorithms rely on probabilistic measurement outcomes, leading to greater numerical instability and requiring many repetitions to provide desired results.
- **Optimized classical algorithms:** Classical algorithms are often highly optimized and tailored to specific problems, incorporating techniques such as heuristics, branch-and-bound methods, and dynamic programming. This presents a major challenge for quantum algorithms: while they may outperform brute-force classical approaches (e.g. Grover's search algorithm can quadratically speed up exhaustive search), they often fail to compete with state-of-the-art classical solvers. While some efforts have been made to integrate such techniques into quantum frameworks – such as branch-and-bound [Mon20] and dynamic programming [Amb+] – adapting these methods to quantum computers remains significantly more difficult than in classical computing.

Unlike for classical algorithms, benchmarking quantum algorithms on real hardware is currently infeasible for relevant problem sizes, as the necessary quantum computers do not yet exist. Simulating quantum computations on classical hardware is also highly limited, as the memory and computational cost grow exponentially with the number of qubits. As a result, many quantum algorithms proposed in recent years claim theoretical speedups, but their practical performance remains unknown. This makes it essential to explore indirect benchmarking methods and develop techniques to estimate or bound their efficiency on potential future quantum hardware. We discuss an approach for this in the next section.

3.4 Benchmarking of quantum algorithms

Currently, no quantum hardware exists that can solve problems at a practically relevant scale. This makes it impossible to directly measure the running time of quantum algorithms. Instead, one has to analyze the structure and complexity of the required quantum circuits and derive theoretical estimates for their performance on future hardware. This section outlines a typical approach to such an analysis.

Gate model

Quantum information is stored in qubits, which are manipulated using quantum gates. The specific set of available gates depends on the architecture of the quantum computer, as well as the encoding and error correction protocols employed. The latter introduces significant overhead, increasing both the number of physical qubits required and the total number of gate operations performed. To ensure a hardware-independent complexity analysis, this thesis considers an idealized model in which error correction overhead is neglected, and only logical qubits and gate operations are taken into account. The analysis assumes access to a small universal gate set, including both single-qubit and two-qubit operations. Since two-qubit gates are typically more difficult to implement and often constitute the main bottleneck, single-qubit gates are disregarded to simplify the analysis. It is important to emphasize that these assumptions favor quantum computing, meaning that the results derived from this approach provide lower bounds rather than accurate estimates of actual resource requirements.

Estimating the quantum gate count

Quantum algorithms often combine classical and quantum components, achieving a speedup through the execution of specific subroutines on a quantum computer. While the overall running time of such hybrid algorithms is difficult to predict due to its dependence on problem instances, it is often possible to estimate the cost of individual quantum subroutines. A common approach to benchmarking is to determine the number of quantum gate operations required for a single execution of a subroutine, considering parameters such as problem size and targeted precision. Since these parameters may vary across subroutine calls, the total cost is estimated using a method known as hybrid benchmarking. This involves classically simulating the full algorithm, tracking the frequency and characteristics of each quantum subroutine call, and summing their respective gate costs.

Chapter 4

Publication: Solving quadratic binary optimization problems using quantum SDP methods: Non-asymptotic running time analysis

This publication builds upon the *Hamiltonian Updates* (HU) algorithm introduced by [GLBKSF22], which was designed to solve the semidefinite programming (SDP) relaxation that arises in the GW algorithm discussed in Section 2.5. The HU algorithm uses certain positive semidefinite (psd) matrices known as *Gibbs states* or *thermal states*, which naturally appear in the formulation of quantum mechanics. While the high-level structure of the HU routine is entirely classical, the computation of Gibbs states can be performed on a quantum computer. By employing quantum algorithms, this process achieves a better asymptotic scaling in the dimension compared to classical methods, potentially leading to a quantum advantage.

However, this improved dimensional scaling comes at the cost of significantly worse dependence on precision. Since the HU algorithm is intended to be used within the GW framework, which includes a rounding step, it is conceivable that lower precision suffices for practical applications. Notably, even in its classical form, the HU algorithm already outperforms standard SDP solvers such as *matrix multiplicative weights* and *interior-point methods* in terms of scaling with dimension, although, again at the expense of less favorable precision scaling.

As discussed in Section 3.3, superior asymptotic scaling does not necessarily translate into a practical quantum advantage for realistically solvable problem instances. The aim of this publication is to rigorously analyze the quantum implementation of the HU algorithm and assess whether a regime of problem instances exists where future quantum hardware could surpass classical computation. Since the original formulation of the HU algorithm was not optimized for non-asymptotic performance, a direct evaluation of its practical feasibility would be insufficient. Consequently, a significant portion of this work is dedicated to improving the algorithm's practical efficiency before benchmarking its performance.

Contents of the publication

The publication presents several contributions:

- A series of improvements to the HU routine, significantly enhancing convergence in numerical benchmarks for both classical and quantum implementations.
- A new infeasibility criterion that substantially reduces the running time for infeasible instances.
- Improved theoretical bounds and a numerical analysis on the precision of the solution returned by the HU algorithm after applying the randomized rounding routine.
- The integration of new quantum algorithms for Gibbs state preparation, improving asymptotic scaling with respect to precision.
- Lower bounds on the number of quantum gate operations required for the HU routine. Based on these bounds, a comparative analysis of the running times of quantum and classical implementations is conducted, with extrapolations to larger problem instances.

Findings and conclusions

Despite the algorithmic improvements introduced in this paper and multiple assumptions favoring the quantum implementation – such as ignoring overhead from error correction and statistical inaccuracies from quantum measurement – the quantum version of HU remains more than ten orders of magnitude away from breaking even with its classical counterpart for tested instances. Furthermore, extrapolations indicate that even for problem instances requiring over a century to solve classically, the quantum speedup remains insufficient, with the performance gap still exceeding seven orders of magnitude.

4.1 Contributions to the publication

The improvements to the HU algorithm were developed and evaluated by me, with input from the other authors. Similarly, I derived and applied the proofs presented in the paper, as well as the bounds used for gate counting. I wrote the majority of the manuscript, excluding the abstract, substantial portions of the introduction, and the conclusion. Additionally, I implemented the complete codebase for generating and benchmarking problem instances, along with parts of the code used to visualize the results.

Solving quadratic binary optimization problems using quantum SDP methods: Non-asymptotic running time analysis

Fabian Henze*[1], Viet Tran[2], Birte Ostermann[3], Richard Kueng[2], Timo de Wolff[3] and David Gross[1]

[1]Institute for Theoretical Physics, University of Cologne, Germany
[2]Institute for Integrated Circuits and Quantum Computing, JKU Linz, Austria
[3]Institute for Analysis and Algebra, TU Braunschweig, Germany

Abstract

Quantum computers can solve semidefinite programs (SDPs) using resources that scale better than state-of-the-art classical methods as a function of the problem dimension. At the same time, the known quantum algorithms scale very unfavorably in the precision, which makes it non-trivial to find applications for which the quantum methods are well-suited. Arguably, precision is less crucial for SDP relaxations of combinatorial optimization problems (such as the Goemans-Williamson algorithm), because these include a final rounding step that maps SDP solutions to binary variables. With this in mind, Brandão, França, and Kueng have proposed to use quantum SDP solvers in order to achieve an *end-to-end* speed-up for obtaining approximate solutions to combinatorial optimization problems. They did indeed succeed in identifying an algorithm that realizes a polynomial quantum advantage in terms of its asymptotic running time. However, asymptotic results say little about the problem sizes for which advantages manifest. Here, we present an analysis of the non-asymptotic resource requirements of this algorithm. The work consists of two parts. First, we optimize the original algorithm with a particular emphasis on performance for realistic problem instances. In particular, we formulate a version with adaptive step-sizes, an improved detection criterion for infeasible instances, and a more efficient rounding procedure. In a second step, we benchmark both the classical and the quantum version of the algorithm. The benchmarks did not identify a regime where even the optimized quantum algorithm would beat standard classical approaches for input sizes that can be realistically solved at all. In the absence of further significant improvements, these algorithms therefore fall into a category sometimes called *galactic*: Unbeaten in their asymptotic scaling behavior, but not practical for realistic problems.

*fhenze2@thp.uni-koeln.de

Contents

1 Introduction

1.1 Non-asymptotic analysis of algorithms

Quantum algorithms are typically compared to classical approaches based on their asymptotic behavior. Indeed, the lack of large-scale, fault-tolerant quantum hardware makes a direct comparison of practical performance difficult for the time being.

The danger of this approach lies in the fact that an asymptotic assessment hides constant factors, which can be substantial. Even if an advantageous scaling behavior has been rigorously established, such an analysis does not directly give information about the minimal size of instances for which a practical speed-up actually manifests. One therefore encounters the risk of designing what is sometimes referred to as *galactic algorithms*: Computational methods that do outperform all others, but only at instance sizes that are so enormous that there is no hope that they will ever be practically relevant. It is therefore a timely and important task to find non-asymptotic estimates for the resource use of quantum algorithms [1, 3, 21, 22].

In doing so, one faces the problem that algorithms published in the academic literature are often not optimized for practical performance. Up until recently, such optimizations were not seen as a priority: The description of *any* novel quantum algorithm beyond the well-known classes discovered by Shor [45], Grover [27], as well as Hassidim Harrow and Lloyd [28], displaying some form of quantum advantage, is considered a significant achievement. This, coupled with the fact that scaled-up quantum hardware remains out of reach, means that there has been little incentive for researchers to optimize implementations.

In particular if one expects to find negative results about the practical usability of published quantum algorithms, it is therefore incumbent upon those performing benchmarks, to first expend reasonable efforts to find an optimized implementation. For this reason, a large part of the present work is spent on improving existing algorithms (Secs. 3 – 5).

1.2 Combinatorial optimization

We are concerned here with quadratic unconstrained binary optimization problems, often abbreviated as *QUBOs*:

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{maximize}} \quad & x^T C x = \sum_{i,j=1}^{n} C_{ij} x_i x_j \\ \text{subject to} \quad & x_i = \pm 1 \quad \text{for } i = 1, \ldots, n. \end{aligned} \tag{1}$$

for a symmetric cost matrix $C \in \mathbb{R}^{n\times n}$. Finding the optimizer is NP-hard, as can be seen e.g. by reduction from the problem of finding a maximum cut in a graph (MAXCUT) – one of Karp's 21 NP-complete problems [30].

The seminal work by Goemans-Williamson [26] provides a randomized rounding procedure with rigorous approximation guarantees for MAXCUT, using the semidefinite programming (SDP) relaxation of the original problem. This SDP relaxation of

Problem (1) is given by

$$\begin{aligned} \underset{X\in\mathbb{R}^{n\times n}}{\text{maximize}} \quad & \operatorname{tr}\left(C^T X\right) = \sum_{i,j=1}^{n} C_{ij}X_{ij} \\ \text{subject to} \quad & \operatorname{diag}(X) = 1, X \succeq 0, \end{aligned} \tag{2}$$

where the final constraint demands that X is symmetric ($X^T = X$) and positive semidefinite, or psd for short. The quantitative relation between the original problem (1) and the relaxed version (2) depends on properties of the coefficient matrix C (e.g. whether is it positive in the SDP- or element-wise sense; see, e.g. Refs. [2, 24, 17]). In this paper, we restrict attention to matrices of the form

$$C = \begin{pmatrix} 0 & B \\ B^T & 0 \end{pmatrix}, \tag{3}$$

where $B \in \mathbb{R}^{n/2\times n/2}$. In order to fix a consistent normalization, we will throughout assume that the coefficient matrix is normalized in operator norm (or spectral norm, i.e. the largest singular value), $\|C\| = 1$. For such instances, the value obtained from applying a *randomized rounding* procedure to the solution of (2), that gives a feasible solution for the original problem, is at most a factor of $(\frac{4}{\pi} - 1) \simeq .273$ smaller than the optimal value of the original problem [2, Sec. 4.1]. Remarkably, the relation does not depend on the dimension n.

The relaxed problem (2) can be represented as a semidefinite program (SDP). As such, it can be tackled, e.g. via *interior point methods* or the *ellipsoid method*. These methods, in particular, interior point methods, are effectively solvable, and ellipsoid methods are, moreover, solvable in polynomial time if proper starting criteria are met. While it the polynomial time solvability of SDPs is not fully settled in general yet due to an observation by O'Donnell [36], the polynomial time solvability for SDPs relaxed from QUBOs via Sums of Squares follows from Raghavendra and Weitz [41]. We refer to standard textbooks, such as [11, 13], for further details. In practice, though, SDP solvers perform poorly in the problem size, both in terms of running time and memory requirements, thus even a polynomial advantage might therefore have significant practical impact.

To simplify the comparison between various methods, we state their performance for the particular case where the matrix B is chosen to be a matrix with column sparsity s, where the non-zero entries are i.i.d. Gaussian random variables. We also assume that the exponent of matrix multiplication equals 3, reflecting the scaling in practical applications. Let μ be the desired precision of the SDP solution. Then, the two prevalent approaches to solve the SDP in Eq. (2) have the following asymptotic running times:

(i) *Interior point methods* require $\mathcal{O}(n^4 \log(\mu^{-1}))$ floating-point operations; see, e.g. [32].

(ii) *Matrix multiplicative weight methods* require $\tilde{\mathcal{O}}(\min\{n^{2.5}s\mu^{-2.5}, n^{2.5}s^{0.5}\mu^{-3.5}\})$ floating-point operations; see, e.g. [8].

We note that μ refers to the approximation of the optimal value of the relaxed problem (2), not the original one (1). The latter, being NP-hard to approximate by

the PCP theorem, does of course not admit any polynomial-time approximation under standard complexity-theoretic assumptions.

With this goal in mind, Ref. [25] developed a classical and a quantum *Hamiltonian Update algorithm* for solving the SDP relaxation (2) using resources that scale better than off-the-shelf solvers in the problem size. On the flip side, their algorithm converges very slowly. That is, the proven running time bounds are extremely unfavorable in the precision μ. More precisely, their main theorem is:

Theorem 1 ([25, Thm. 1])**.** *Let C be a (real-valued) symmetric $n \times n$ matrix with column sparsity s. Then the problem* (2) *can be solved up to additive accuracy $n \|C\| \mu$ in running time*

$$\tilde{\mathcal{O}}(n^{1.5}(\sqrt{s})^{1+o(1)}\mu^{-28+o(1)}\exp(1.6\sqrt{12\log(\mu^{-1})}))$$

on a quantum computer and in running time

$$\tilde{\mathcal{O}}(\min\{n^2 s\mu^{-12}, n^3\mu^{-8}\})$$

on a classical computer.

We are thus faced with the following situation: The algorithm of Ref. [25] shows that, for fixed precision μ, quantum computers can, in principle, solve the SDP relaxation (2) with a running time that scales more favorably in the problem size than any known classical approach with rigorous performance guarantees. At the same time, the proven dependency on μ suggests that the theoretical advantage will only manifest for enormously large problem instances, which cannot be practically executed at all.

The purpose of the present paper is to optimize the results of Ref. [25], derive sharper bounds, and finally benchmark the performance of the optimized version, in order to estimate its performance for realistic scenarios.

Another recent follow-up to Ref. [25] is Ref. [10], which explores the use of *iterative refinement* techniques to speed up convergence. We have not included a quantitative comparison between these two approaches in this manuscript, because no reference implementation is available, and we are still in the process of clarifying some questions about algorithmic details with the authors [9].

We refer to Refs. [16, 15, 4, 5] for more details on quantum algorithms for solving SDPs.

1.3 Organization of the paper

The structure of the paper is as follows:

In Sec. 2, we provide an overview of the Hamilton Updates (HU) algorithm introduced in Ref. [25]. Sec. 3 presents several non-asymptotic improvements to HU that enhance both its classical and quantum performance. In particular: Secs. 3.1-3.4 detail methods for reducing the number of iterations needed in practice to find a feasible solution or to certify infeasibility. In Sec. 3.5, we numerically compare the performance of the improved algorithm to the original one. Sec. 4.1 provides an asymptotically tighter bound on the precision of the objective value after rounding, improving upon the results in Ref. [25]. Complementing this, Sec. 4.2 numerically examines the behavior

of the precision after rounding. Sec. 5 presents an improved quantum subroutine that uses algorithms with better dependence on the precision for Gibbs state preparation and Hamiltonian simulation, offering improvements over Ref. [25]. Sec. 6 provides an overview of the analytical and numerical results from Secs. 4 and 5. Finally, in Sec. 7, we benchmark the improved algorithm by running it classically and estimating the minimum number of gates required for a quantum implementation based on the constructions in Refs. [34, 5].

1.4 Notation

The following table summarizes the notational conventions we use throughout the manuscript:

Symbol	Description	Definition
$A \preceq B$	positive semidefinite (psd) order	$A \preceq B \Leftrightarrow A, B$ symmetric, and $x^T A x \leq x^T B x \ \ \forall x \in \mathbb{R}^n$
$\|x\|_{\ell_1}$	ℓ_1 norm	$\|x\|_{\ell_1} = \sum_i \|x_i\|$
$\|A\|$	operator (or Schatten-∞) norm	$\|A\| = \sup_{x \in \mathbb{R}^n : \|x\|=1} \{\|Ax\|\}$
$\|A\|_{\mathrm{tr}}$	trace (or Schatten-1) norm	$\|A\|_{\mathrm{tr}} = \mathrm{tr}(\|A\|), \quad \|A\| = \sqrt{A^T A}$
$\|A\|_{\max}$	max (or ℓ_∞) norm	$\|A\|_{\max} = \max_{ij} \|A_{ij}\|$
$\|A\|_{\ell_1 \to \ell_2}$	$\ell_1 \to \ell_2$ norm	$\|A\|_{\ell_1 \to \ell_2} = \sup_{x \in \mathbb{R}^n : \|x\|_{\ell_1}=1} \{\|Ax\|\}$
$\|A\|_{\mathrm{tr} \to \mathrm{tr}}$	trace-to-trace norm	$\|A\|_{\mathrm{tr} \to \mathrm{tr}} = \sup_{B \in \mathbb{R}^{n \times n} : \|B\|_{\mathrm{tr}}=1} \{\|A(X)\|_{\mathrm{tr}}\}$
$\|A\|_{\infty \to \infty}$	infinity-to-infinity norm	$\|A\|_{\infty \to \infty} = \sup_{B \in \mathbb{R}^{n \times n} : \|B\|=1} \{\|A(X)\|\}$
$[n]$	index set	$[n] = \{i \in \mathbb{N} \| 1 \leq i \leq n\}$
$R(\rho\|\sigma)$	quantum relative entropy	$S(\rho\|\sigma) = \mathrm{tr}\left(\rho(\ln \rho - \ln \sigma)\right)$ with $\rho, \sigma \succeq 0, \mathrm{tr}(\rho) = \mathrm{tr}(\sigma) = 1$
C	normalized cost matrix	$C \in \mathbb{R}^{n \times n}, C^T = C, \|C\| = 1$
x	solution for QUBO (1)	$x \in \{-1, 1\}^n$
X	solution for SDP (2)	$X \in \mathbb{R}^{n \times n}$, X is psd, $\mathrm{diag}(X) = \mathbb{1}$
H	Hamiltonian and (potential) solution for (8)	$H \in \mathbb{R}^{n \times n}, H^T = H$
ρ or ρ_H	Gibbs state and (potential) solution for (6)	$\rho_H = \exp(-H)/\mathrm{tr}(\exp(-H))$
γ	threshold for objective value	Defined in (8).
n	dimension of cost matrix	
s	(column) sparsity	maximum number of non-zero entries per column in a matrix
μ	SDP precision	Let $X^\star$ be an optimal solution to (2). A solution X to (2) has precision μ if $\mathrm{tr}(X^\star C) - \mathrm{tr}(XC) \leq n\mu$.
ϵ	precision of HU constraints	Defined in (8).
ν	precision after randomized rounding	Defined in (42).
ΔH	matrix for HU (cost/diagonal) update	$H \mapsto H + \lambda \Delta H$, defined in (20)
λ or λ_c, λ_d	step size/length for HU (cost/diagonal) update	
P_c	matrix for cost update	$P_c = \gamma \mathbb{1} - C$
$P_d^{\ell_1}$	matrix for diagonal update with ℓ_1 norm	$P_d^{\ell_1} = \mathrm{sign}(\mathrm{diag}(\rho) - \mathbb{1}/n) - \mathrm{tr}(\mathrm{sign}((\mathrm{diag}\,\rho) - \mathbb{1}/n))/n\mathbb{1}$
$P_d^{\ell_2}$	matrix for diagonal update with ℓ_2 norm	$P_d^{\ell_2} = (\mathrm{diag}(\rho) - \mathbb{1}/n)/\max_i \|\rho_{ii} - 1/n\|$
$M^{(k)}$	momentum in k^{th} iteration	$M^{(k)} = \lambda_{c/d}(\Delta H)^{(k)}$
β	momentum hyperparameter	$0 \leq \beta < 1$
F	free energy	$F(H) = -\ln(\mathrm{tr}(\exp(-H)))$
b	number of qubits used to store H	

Table 1: Summary of notation used in this paper.

2 The Hamiltonian Update algorithm

Now, we give a summary of the Hamiltonian Updates algorithm for solving the SDP in Eq. (2). Compared to the original presentation of Ref. [25], we slightly modify the notation involving the update step, in a way that facilitates stating the improvements in Sec. 3. We give the high-level algorithm in Alg. 1 and an illustration in Fig. 1.

2.1 Reduction to feasibility problems

We express the HU algorithm as an optimization over the renormalized psd matrix

$$\rho = \frac{1}{n} X. \tag{4}$$

The two representations are obviously equivalent – but the convention adopted here will later allow us to formulate a quantum version, where ρ will be a *density matrix* (i.e. $\rho \succeq 0$ and $\mathrm{tr}(\rho) = 1$) describing the state of a physical quantum system. In particular, the constraint on the diagonal now reads $\mathrm{diag}(\rho) = \frac{1}{n}\mathbb{1}$.

Using the matrix Hölder inequality, we can then bound the objective function of the SDP relaxation:

$$|\mathrm{tr}\,(C\rho)| \leq \|C\|\,\|\rho\|_{\mathrm{tr}} = \|C\|\,\mathrm{tr}(\rho) = 1,$$

because the trace norm of a psd matrix is equal to its trace. This ensures that the optimal value of the re-scaled version of SDP (2) falls into the interval $[-1, 1]$.

In a next step, we will transform the optimization problem into a series of feasibility problems. Choose a precision parameter ϵ_b and perform a binary search over the interval $[-1, 1]$, to find a value $\gamma^\star$ such that the program

$$\begin{aligned} \underset{\rho \in \mathbb{R}^{n\times n}}{\text{find}} \quad & \rho \succeq 0, \quad \mathrm{tr}(\rho) = 1 \\ \text{subject to} \quad & \gamma - \mathrm{tr}(C\rho) \leq 0 \\ \text{and} \quad & \sum_i \left|\rho_{ii} - \frac{1}{n}\right| = 0 \end{aligned} \tag{5}$$

is feasible for $\gamma = \gamma^\star$, but not for $\gamma = \gamma^\star + \epsilon_b$. A binary search finds such a value in $\mathcal{O}(\log_2(\epsilon_b^{-1}))$ iterations. Therefore, the feasibility problems (5) can find the optimal value of (2) with an overhead that is logarithmic in the desired precision.

However, even the feasibility problem cannot be decided by a practical algorithm for exact constraints. Let ϵ_f be another precision parameter. We say that ρ is ϵ_f*-feasible* for the program (5) if it satisfies

$$\begin{aligned} \underset{\rho \in \mathbb{R}^{n\times n}}{\text{find}} \quad & \rho \succeq 0, \quad \mathrm{tr}(\rho) = 1 \\ \text{subject to} \quad & \gamma - \mathrm{tr}(C\rho) < \epsilon_f \\ \text{and} \quad & \sum_i \left|\rho_{ii} - \frac{1}{n}\right| < \epsilon_f. \end{aligned} \tag{6}$$

The program (5) is ϵ_f*-feasible* if an ϵ_f-feasible ρ exists.

In the section below, we will introduce the *Hamiltonian Update* (HU) routine for solving Eq. (6). It satisfies the following conditions:

1. If (5) is feasible, the HU routine outputs an ϵ_f-feasible solution ρ.

2. If (5) is not ϵ_f-feasible, the HU routine outputs no solution.

3. If (5) is ϵ_f-feasible but not feasible, the HU routine outputs either an ϵ_f-feasible solution or no solution.

Thus, conversely,

1. If the HU algorithm succeeds, it will output an ϵ_f-feasible $\rho^\star$. In Sec. 4, we will describe *rounding algorithms*, which construct exact solutions given $\rho^\star$.

2. If the HU algorithm fails, we know that (5) is not strictly feasible.

Therefore, a binary search using the HU algorithm will output an ϵ_f-feasible solution $\rho^\star$ with objective value $\gamma^\star$, such that (5) is not strictly feasible for $\gamma = \gamma^\star + \epsilon_b$.

For simplicity, in what follows, we will restrict to the case where $\epsilon_f = \epsilon_b$, and denote this common precision parameter by ϵ.

2.2 The Hamiltonian Update step

The idea behind Hamiltonian Updates is to express ρ as a *Gibbs states*, i.e. one writes

$$\rho_H = \frac{\exp(-H)}{\mathrm{tr}(\exp(-H))} \tag{7}$$

for some symmetric matrix $H \in \mathbb{R}^{n\times n}$. We refer to H as the *Hamiltonian*, in accordance with standard physics terminology. This way, the constraints $\rho \succeq 0$ and $\mathrm{tr}(\rho) = 1$ are automatically satisfied. Thus, the problem reduces to finding an H such that the constraints on ρ_H are fulfilled:

$$\begin{aligned} \underset{\substack{H\in\mathbb{R}^{n\times n}\\ H=H^T}}{\text{find}} \quad & H, \\ \text{subject to} \quad & \gamma - \mathrm{tr}(C\rho_H) < \epsilon \\ \text{and} \quad & \sum_i \left|(\rho_H)_{ii} - \frac{1}{n}\right| < \epsilon \end{aligned} \tag{8}$$

To find a suitable Hamiltonian, we start with the $n \times n$ zero matrix $H = 0$, and refine it in a series of *update steps*, described next.

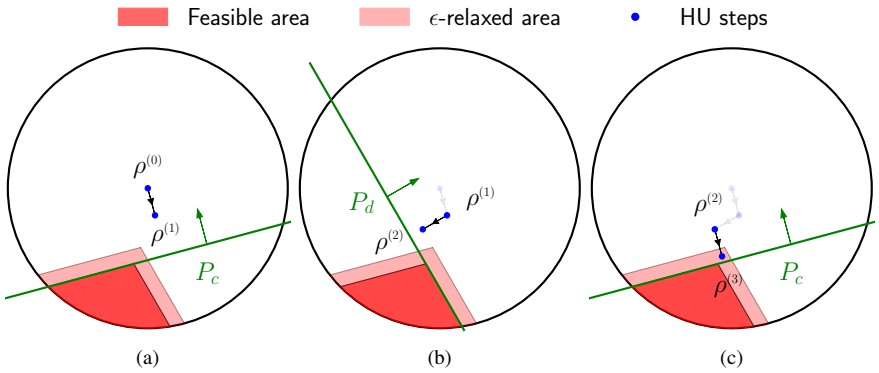

Figure 1: *Illustration of the Hamiltonian Updates algorithm.* The circle depicts the space of trace-one psd matrices, with ρ_0 lying in the center. The feasible region is shown in dark red, the ϵ-feasible region is marked light red. Each graphic shows a single iteration of HU: At the start of each update, a matrix P is calculated that defines a hyperplane (shown in green) separating the current ρ and the feasible region. By updating $\rho_H \to \rho_{H+\lambda P}$ (i.e. penalizing infeasible directions), ρ moves towards the hyperplane. For simplicity this is depicted by a straight line. This is generally not the case, as ρ depends non-linearly on H. The procedure ends when ρ enters the ϵ-feasible region (i.e. all constraints are fulfilled up to precision ϵ as defined in (8)).

Algorithm 1 Simplified Hamiltonian Updates

Require: Cost matrix C, threshold objective value γ, precision parameter ϵ

Ensure:

Condition	Output
(5) is feasible	an ϵ-feasible ρ
(5) is not ϵ-feasible	false
else	undefined (an ϵ-feasible ρ or false)

function HAMILTONIAN_UPDATES(C, γ, ϵ)
  $H \leftarrow 0_{n\times n}, \rho \leftarrow \mathbb{1}/n$ and $F \leftarrow -\ln(n)$
  while $F \leq 0$ **do** ▷ Main loop of HU
    if ρ is ϵ-feasible **then**
      return ρ
    else
      compute ΔH ▷ Computed from the violations of ρ
      $H \leftarrow H + \lambda\Delta H$ ▷ Update Hamiltonian
      $\rho \leftarrow \exp(-H)/\operatorname{tr}(\exp(-H))$ ▷ Update Gibbs state
      $F \leftarrow -\ln(\operatorname{tr}(\exp(-H)))$ ▷ Update free energy
    end if
  end while
  return false ▷ $F > 0 \rightarrow$ No feasible solution exists
end function

To keep notation succinct, we usually make the dependence of ρ_H on H implicit, and write $\rho = \rho_H$ if the Hamiltonian is clear from context. Let $\tilde{\rho} = \operatorname{diag}(\rho)$ be the diagonal matrix with $\tilde{\rho}_{ii} = \rho_{ii}$. We define the two symmetric matrices:

$$P_c = \gamma\mathbb{1} - C, \tag{9}$$

$$P_d^{\ell_1} = \operatorname{sign}(\tilde{\rho} - \mathbb{1}/n) - \operatorname{tr}(\operatorname{sign}(\tilde{\rho} - \mathbb{1}/n))/n\mathbb{1}, \tag{10}$$

where the sign function is applied element-wise to the matrices. (The origin of the superscript ℓ_1 for P_d will become clear in Sec. 3.2).

The matrices P_c and $P_d^{\ell_1}$ allow us to reformulate the two feasibility constraints (8) as

$$\gamma - \operatorname{tr}(C\rho) = \operatorname{tr}(P_c\rho) < \epsilon \tag{11}$$

$$\sum_i |\rho_{ii} - 1/n| = \operatorname{tr}(P_d^{\ell_1}\rho) < \epsilon. \tag{12}$$

While the correctness of (11) is easy to see, (12) needs a short calculation to confirm:

$$\begin{aligned}\sum_i |\rho_{ii} - 1/n| &= \sum_i (\rho_{ii} - 1/n)\operatorname{sign}(\rho_{ii} - 1/n) \\ &= \sum_i \rho_{ii}\operatorname{sign}(\rho_{ii} - 1/n) - \sum_i \operatorname{sign}(\rho_{ii} - 1/n)/n \\ &= \operatorname{tr}(\rho \operatorname{sign}(\tilde{\rho} - \mathbb{1}/n)) - \operatorname{tr}(\rho)\operatorname{tr}(\operatorname{sign}(\tilde{\rho} - \mathbb{1}/n)/n) \\ &= \operatorname{tr}(\rho \operatorname{sign}(\tilde{\rho} - \mathbb{1}/n)) - \operatorname{tr}(\rho \operatorname{tr}(\operatorname{sign}(\tilde{\rho} - \mathbb{1}/n))/n) \\ &= \operatorname{tr}(\rho P_d^{\ell_1}),\end{aligned} \tag{13}$$

where we used $\operatorname{tr}(\rho) = 1$ in the third step.

The matrices P_c and $P_d^{\ell_1}$ can be interpreted as normal vectors for hyperplanes in the vector space of symmetric $n \times n$ matrices which we also endow with the trace (or Frobenius) inner product $A, B \mapsto \operatorname{tr}(AB)$. The hyperplanes separate the current iterate ρ from the feasible region defined in the feasibility SDP (8). The trace inner products $\operatorname{tr}(P_c\rho)$ and $\operatorname{tr}(P_d^{\ell_1}\rho)$ measure how strongly the constraints are violated. If both constraints are violated by less than ϵ, the algorithm stops and outputs the solution ρ. Otherwise, in order to reduce the violations, HU applies *cost updates* and *diagonal updates*, defined, respectively, as

$$H \mapsto H + \lambda \Delta H, \quad \text{where} \quad \Delta H = \begin{cases} P_c & \text{during cost updates,} \\ P_d^{\ell_1} & \text{during diagonal updates.} \end{cases} \tag{14}$$

for some step size λ.

By adding $\lambda\Delta H$ to the current Hamiltonian H, the updated Gibbs state

$$\rho_{H+\lambda\Delta H} = \exp(-(H + \lambda\Delta H))/\operatorname{tr}(\exp(-(H + \lambda\Delta H)))$$

will move towards to the separating plane (or even cross it when λ is too large). Note that P_c is constant given γ, while $P_d^{\ell_1}$ depends on ρ and therefore changes with each update. We will discuss quantitative bounds on the distance to the feasible region in Sec. 3.4. The algorithm then iterates this Hamiltonian update procedure many times to get closer and closer to the feasible region – hence the name.

Stated as is (and how it was proposed in Ref. [25]), this meta-algorithm faces two major problems which renders it inefficient in practice: Firstly, an analytical running time analysis provided in Sec. 3.4.2 shows that the required number of iterations depends quadratically on the precision ϵ. Secondly, the Hamiltonian Updates algorithm detects infeasible instances by checking if an *a priori* upper bound on the number of iterations has been reached. This turns out to be rather inefficient in practice. In this work, we provide substantial improvements that address both of these problems. This is the content of the next section.

3 Improved convergence for Hamiltonian Updates

In this section, we introduce several improvements to the Hamiltonian Updates meta-algorithm, which benefit both the classical and the quantum version. In Secs. 3.2-3.3, we present heuristics designed to reduce the number of iterations required for the

algorithm to converge. Sec. 3.4 gives improved estimation techniques for the change in relative entropy used to prove infeasibilty of a given problem instance. The complete enhanced algorithm is outlined as Alg. 2 and 3. Finally, in Sec. 3.5 we compare these improvements numerically against the original algorithm.

Algorithm 2 Improved Hamiltonian Updates

Require: Normalized cost matrix C, threshold objective value γ, precision parameter ϵ, initial step lengths λ_c and λ_d, momentum hyperparameter β

Ensure:

Condition	Output
(5) is feasible	an ϵ-feasible ρ
(5) is not ϵ-feasible	false
else	undefined (an ϵ-feasible ρ or false)

function HAMILTONIAN_UPDATES($C, \gamma, \epsilon, \lambda_c, \lambda_d, \beta$)
  $P_c \leftarrow -C + \gamma \mathbb{1}$
  $H \leftarrow 0_{n\times n}$
  $M \leftarrow 0_{n\times n}$
  $F = -\ln(n)$
  $\rho \leftarrow \frac{1}{n}\mathbb{1}_{n\times n}$

  while $F \leq 0$ **do** ▷ Main loop of HU
    if $\mathrm{tr}(P_c\rho) > \epsilon$ **then**
      $\Delta H \leftarrow \mathrm{tr}(P_c\rho)P_c + \frac{\beta}{\lambda_c}M$
      $H, \rho, F, \lambda_c \leftarrow$ UPDATE$(H, \Delta H, \lambda_c)$ ▷ Apply cost update
      $M \leftarrow \lambda_c \Delta H$ ▷ Update momentum

    else if $\sum_i |\rho_{ii} - 1/n| > \epsilon$ **then**
      $P_d^{\ell_2} \leftarrow (\mathrm{diag}(\rho) - \mathbb{1}/n)/\max_i |\rho_{ii} - 1/n|$
      $\Delta H \leftarrow P_d^{\ell_2} + \frac{\beta}{\lambda_d}M$
      $H, \rho, F, \lambda_d \leftarrow$ UPDATE$(H, \Delta H, \lambda_d)$ ▷ Apply diag. update
      $M \leftarrow \lambda_d \Delta H$ ▷ Update momentum

    else
      return ρ ▷ ρ is ϵ-feasible
    end if

  end while
  return false ▷ $F > 0 \rightarrow$ No feasible solution exists
end function

Algorithm 3 Update function for Hamiltonian Updates

Require: Hamiltonian H, update matrix ΔH, current step length λ_c or λ_d

Ensure: updated Hamiltonian H_{new}, current Gibbs state ρ, current free energy F, updated step length λ_c or λ_d

function UPDATE($H, \Delta H, \lambda$)
  $H_{\text{new}} \leftarrow H + \lambda \Delta H$ ▷ Compute new Hamiltonian
  compute $\exp(-H_{\text{new}})$ ▷ Compute matrix exponential for ρ and F
  $\rho_{\text{new}} \leftarrow \exp(-H_{\text{new}}) / \operatorname{tr}(\exp(-H_{\text{new}}))$

  while $\operatorname{tr}(\Delta H \rho_{\text{new}}) < 0$ **do**: ▷ Check for *overshoots*
    $\lambda \leftarrow 0.5\lambda$ ▷ Reduce step size
    $H_{\text{new}} \leftarrow H + \lambda \Delta H$ ▷ Re-compute new Hamiltonian
    compute $\exp(-H_{\text{new}})$
    $\rho_{\text{new}} \leftarrow \exp(-H_{\text{new}}) / \operatorname{tr}(\exp(-H_{\text{new}}))$
  end while

  $F \leftarrow -\ln(\operatorname{tr}(\exp(-H_{\text{new}})))$ ▷ Compute free energy
  $\lambda \leftarrow 1.3\lambda$ ▷ Increase step size for the next iteration

  return $H_{\text{new}}, \rho_{\text{new}}, F, \lambda$
end function

3.1 Adaptive step length

Recall from Eq. (14) above, that the individual updates of the Hamiltonian $H \in \mathbb{R}^{n \times n}$ take the following form:

$$H \mapsto H + \lambda P, \tag{15}$$

where P can be either P_c or $P_d^{\ell_1}$. As is commonly the case for iterative algorithms, there is a trade-off in choosing the *step size* or *learning rate* λ: Small choices of λ mean that violations of the constraints take many iterations to be corrected, while too large values of λ increase the danger of *overshooting*. In our case, overshooting corresponds to moving to the other side of the separating hyperplane, in which case we do not have a guaranteed improvement anymore (c.f. Sec. 3.4.1).

The algorithm of Ref. [25] uses a constant step length $\lambda = \epsilon/16$ that only depends on the desired target accuracy ϵ. In contrast, here we propose to use two *adaptive step lengths*

$$\lambda_c \qquad \text{and} \qquad \lambda_d,$$

one for each constraint in (8). We choose the concrete step lengths according to the following heuristic: The algorithm maintains the current step sizes λ_c and λ_d, which

are increased by a constant factor after each corresponding update – we find that multiplying with 1.3 works well in practice. We say that the algorithm has *overshot* if, after an update, the sign of the constraint has reversed, i.e. if $\mathrm{tr}(P\rho_{H+\lambda_{c/d}P}) < 0$. This sign is checked after every update. In case an overshot did occur, λ is halved, and ρ is re-computed for the now smaller step size (c.f. Alg. 3).

This approach means that if an overshoot occurs, we must recompute the Gibbs state, which incurs an overhead in computation time. Hence, the possibility of overshooting manifests itself in a slight increase of the average computation time per iteration, when compared to the constant step length method. However, as we demonstrate numerically in Sec. 3.5, this is more than compensated for by an overall faster step-wise progress which significantly reduces the total number of iterations required, especially in the early phase of the HU meta-algorithm.

3.2 Euclidean-norm based P_d

In the original Ref. [25], the authors address the violation of the diagonal constraint in (8) using a matrix $P_d^{\ell_1}$ that corresponds to the ℓ_1 norm: $\mathrm{tr}(P_d^{\ell_1}\rho) = \sum_i |\rho_{ii} - 1/n|$ (see Eq. (10)). This approach is a natural choice, as this norm reflects the feasibility constraint in Eq. (8) which is also formulated in terms of the ℓ_1 norm. However, closer inspection reveals that the correction provided by $P_d^{\ell_1}$ may be suboptimal. After all, it only considers the sign of the deviations in each entry, not their magnitude.

We propose a new approach where P_d is proportional to the violation in each component. To achieve this, we modify (10) by removing the sign function. Because the trace term in (10) becomes zero under this modification, the result is

$$P_d^{\ell_2} = \tilde{\rho} - \mathbb{1}/n. \tag{16}$$

The trace with the modified matrix evaluates to the squared Euclidean or ℓ_2 norm of the deviation:

$$\begin{aligned} \mathrm{tr}(\rho P_d^{\ell_2}) &= \mathrm{tr}(\rho(\tilde{\rho} - 2\mathbb{1}/n)) + \mathrm{tr}(\rho\mathbb{1}/n) \\ &= \mathrm{tr}(\rho(\tilde{\rho} - 2\mathbb{1}/n)) + 1/n \\ &= \sum_i \left(\rho_{ii}^2 - 2\rho_{ii}/n + 1/n^2\right) \\ &= \sum_i (\rho_{ii} - 1/n)^2. \end{aligned} \tag{17}$$

Note that this is a much more common choice as a loss function in gradient descent algorithms. To further optimize and to improve numerical stability, one can experiment with different normalizations for $P_d^{\ell_2}$. Although the differences are generally minor, because the adaptive step size defined in the previous section adjusts well to different normalizations, we find experimentally that dividing $P_d^{\ell_2}$ by its maximum absolute entry yields the best results.

Additionally, we scale P_c in each update with the corresponding distance $\mathrm{tr}(P_c\rho)$. Thus, the new matrix

$$\tilde{P}_c = \mathrm{tr}(P_c\rho)P_c, \tag{18}$$

corresponds to larger corrections in the cost update when ρ is further away from the feasible region.

In Fig. 3, we compare the performance of HU using $P_d^{\ell_2}$ instead of $P_d^{\ell_1}$, observing that this modification results in a speedup of approximately a factor of two to three.

3.3 Adding a momentum term

In gradient descent methods, it is common to also add a so-called *momentum term* [38], which often empirically increase the speed of convergence.

In the following, we use a superscript $(\cdot)^{(k)}$ to refer to the value of a variable in the k^{th} iteration. Define the *momentum term* to be

$$M^{(k)} = \begin{cases} \lambda_c^{(k)}(\Delta H)^{(k)} & \text{for cost update in } k^{\text{th}} \text{ step,} \\ \lambda_d^{(k)}(\Delta H)^{(k)} & \text{for diag. update in } k^{\text{th}} \text{ step,} \end{cases}$$
$$M^{(0)} = 0.$$

Next, choose a new hyperparameter $\beta \in (0, 1)$ and modify the update rule (14) to read

$$H^{(k+1)} = \begin{cases} H^{(k)} + \lambda_c^{(k)}(\Delta H)^{(k)} & \text{for cost update in } k^{\text{th}} \text{ step,} \\ H^{(k)} + \lambda_d^{(k)}(\Delta H)^{(k)} & \text{for diag. update in } k^{\text{th}} \text{ step,} \end{cases} \tag{19}$$

$$(\Delta H)^{(k)} = \begin{cases} \left(\tilde{P}_c\right)^{(k)} + \frac{\beta}{\lambda_c^{(k-1)}} M^{(k-1)} & \text{for cost update in } k^{\text{th}} \text{ step,} \\ \left(P_d^{\ell_2}\right)^{(k)} + \frac{\beta}{\lambda_d^{(k-1)}} M^{(k-1)} & \text{for diag. update in } k^{\text{th}} \text{ step.} \end{cases} \tag{20}$$

Numerically, we find that a values of β between 0.4 and 0.5 achieve the best results, with reductions in the number of iterations by roughly 30-40% (c.f. Fig. 3 and Tab. 2).

3.4 Free energy tracking

When we cannot guarantee that a particular SDP instance is feasible, it is crucial to find a *termination criterion*; otherwise, the HU routine would run indefinitely. This can be achieved by bounding the *quantum relative entropy*

$$R(\rho^\star \| \rho) = \operatorname{tr}\left(\rho^\star(\log \rho^\star - \log \rho)\right) \tag{21}$$

between any solution $\rho^\star$ (assuming that one exists) and the current state ρ. For properties of the quantum relative entropy, we refer to standard textbooks, e.g. Ref. [43].

It is known that the relative entropy distance between the maximally mixed state $\rho_0 = \mathbb{1}/n$ and any other state is upper-bounded by $\ln(n)$. We demonstrate in this section that one can lower-bound the decrease in relative entropy distance between the current Gibbs state ρ and $\rho^\star$ (if it exists) in each update. By choosing the maximally mixed state as the initial state ρ_0, we are guaranteed that the cumulative change in relative entropy cannot exceed $\ln(n)$, if a solution does indeed exist. By the same

token, if the estimate of total relative entropy distance reduction exceeds $\ln(n)$, it is certain that a feasible solution does not exist.

This "relative entropy tracking" procedure serves two different purposes: First, if the algorithm detects that it would have covered a relative entropy distance of $\ln(n)$, but has not yet found a solution, we know that the problem is infeasible. Using terminology standard in quantum information, we could call this a *heralded* event: Meaning that if it occurs, we can draw rigorous conclusions from it, but, at the beginning of the algorithm, it is unclear after how many iterations it will be detected. Second, we would like to have an *a priori* upper bound on the number of iterations required before infeasibility is detected.

Ref. [25] uses a single bound to serve both these purposes. In contrast, we report a large gain in practical performance by using different estimates for the two goals. In Sec. 3.4.1, we introduce a new method for tracking the entropy change that makes use of another quantity from statistical mechanics called *free energy*. We observe numerically that this new approach can achieve speedups over the original method of a factor > 10000 (c.f. Tab. 2).

Our a priori bound on the number of iterations is presented in Section 3.4.2. Compared to Ref. [25], it includes a treatment of the momentum term (c.f. Section 3.3) and it improves the estimate by a constant factor. The theoretical guarantee does not cover all heuristics we employ, in particular it does not take adaptive step sizes and the improved way of choosing P_d into account. We re-iterate that these are pessimistic worst-case bounds and that the observed practical performance is much better.

3.4.1 Termination criterion

On a high level, our improved relative entropy tracking method exploits the observation that the decrease in distance to the feasible set is larger, the further away the current state is from being feasible. In contrast, [25] uses a worst-case bound that does not take into account the faster decrease of the relative entropy distance that happens especially in the early steps of the algorithm.

We now show how the relative entropy R can be estimated during the HU routine. We can rewrite the definition of the relative entropy (21) as

$$R(\rho^\star||\rho) = \mathrm{tr}(\rho^\star \ln(\rho^\star)) + \mathrm{tr}(\rho^\star H) + \ln(\mathrm{tr}(\exp(-H))). \tag{22}$$

In statistical mechanics,

$$F(H) = -\ln(\mathrm{tr}(\exp(-H))) \tag{23}$$

is called the *free energy* (at inverse temperature 1). Initially, for $H_0 = 0$ we have $F(0) = -\ln(n)$. Then, the total change in relative entropy with respect to the initial Gibbs state $\rho_0 = \mathbb{1}/n$ is given by

$$\Delta R(\rho_0, \rho) = R(\rho^\star||\rho) - R(\rho^\star||\rho_0) = \mathrm{tr}(\rho^\star H) - F(H) - \ln(n). \tag{24}$$

In the HU algorithm, the Hamiltonian H in the k^{th} iteration is of the form

$$H^{(k)} = \sum_{k'=1,\ldots,k} c_{k'} P^{(k')},$$

with coefficients $c_{k'} > 0$. By construction, the matrices $P^{(k')}$ penalize infeasible directions and thus fulfil $\mathrm{tr}(\rho^\star P^{(k')}) \leq 0$. Then, we always have $\mathrm{tr}(\rho^\star H) \leq 0$ and therefore,

$$\Delta R(\rho_0, \rho) \leq -F(H) - \ln(n). \tag{25}$$

Next, we use that the absolute change in relative entropy distance is upper bounded by $\ln(n)$ and therefore

$$-\ln(n) \leq \Delta R(\rho_0, \rho) \leq -F(H) - \ln(n). \tag{26}$$

Thus, we know that if a feasible solution $\rho^\star$ exists, we always have

$$F(H) \leq 0 \tag{27}$$

for any H occurring as part of the HU routine. Hence, the task of tracking the relative entropy translates into evaluating the free energy. Classically, this quantity can be easily computed with no relevant additional computational effort, as one already has to compute $\exp(-H)$ for the Gibbs state in each iteration.

On a quantum computer, we instead bound this quantity indirectly via its derivative. For this, consider a single update. Let H be the initial Hamiltonian, and $H + \lambda\Delta H$ the one after the HU step. The change in free energy is

$$\Delta F = F(H + \lambda\Delta H) - F(H), \tag{28}$$

and its derivative with respect to λ is given by the expectation value of the update term $\lambda\Delta H$ with respect to the final state:

Lemma 2. *For all symmetric $H, \Delta H \in \mathbb{R}^{n\times n}$ and $\lambda \in \mathbb{R}$, the free energy satisfies*

$$\partial_\lambda F(H + \lambda\Delta H) = \mathrm{tr}(\rho_{H+\lambda\Delta H}\lambda\Delta H). \tag{29}$$

The proof is given in Appendix A.1.

Now, by integrating both sides in (29), the change in free energy is given by:

$$\Delta F = \int_0^\lambda \mathrm{tr}(\rho_{H+\lambda'\Delta H}\lambda'\Delta H)\,\mathrm{d}\lambda'. \tag{30}$$

It is known that the function

$$\lambda \mapsto F(H + \lambda\Delta H)$$

is concave (a fact sometimes referred to as *Bogoliubov inequality* [6, Lem. 2] in quantum statistical mechanics). Thus, we can bound ΔF by evaluating $\mathrm{tr}(\rho_{H+\lambda\Delta H}\lambda\Delta H)$ for multiple values $0 < \lambda' \leq \lambda$ and computing

$$\Delta F \geq \sum_{\lambda'_i} (\lambda'_i - \lambda'_{i-1})\,\mathrm{tr}(\rho_{H+\lambda'_i\Delta H}\Delta H). \tag{31}$$

These expected values can be computed relatively cheaply on a quantum computer (compared to the cost of estimating the diagonal of ρ, see Sec. 5).

We have studied the behavior of the improved termination criterion numerically. The results are shown in Fig. 2.

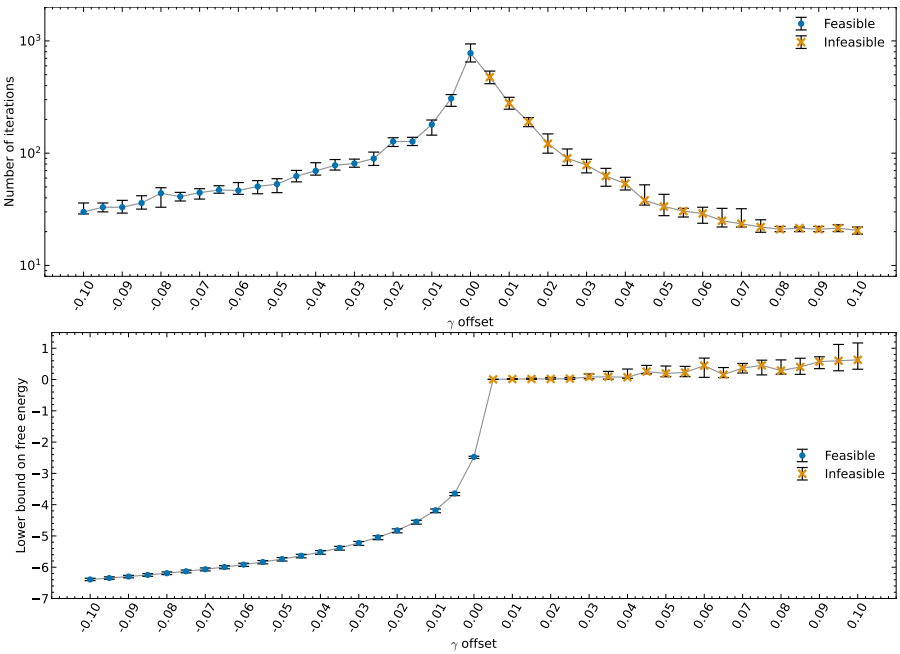

Figure 2: Behavior of the improved HU algorithm on an instance used with parameters $n = 1024$, $s = 16$ and $\epsilon = 0.001$, generated as described in Sec. 3.5. The algorithm terminated after finding an ϵ-feasible solution for $\gamma := \gamma^\star + \text{offset}$, or once the free energy became positive. Here, $\gamma^\star$ is the optimal objective value determined by an SDP solver [37]. The error bars show the upper and lower quartiles.
Upper panel: Number of iterations required until convergence as a function of the γ-offset. The improved termination criterion comes into play on the right tail of the curve. We observe that the number of iterations required to certify infeasibility goes down the further the problem specification is from a feasible one. (This contrasts to the fixed termination criterion used in [25]).
Lower panel: The lower bound on the free energy at time of termination. We observe numerically that the bound increases as γ gets closer to the optimal value. Whether this effect can be exploited for algorithmic improvements is a question we leave open.

3.4.2 Convergence guarantee

We now provide an *a priori* bound on the maximum number of steps required for the algorithm to find a feasible solution, assuming that one does exist. The result goes beyond Ref. [25] in two ways: It includes a treatment of the momentum term (c.f. Section 3.3) and it improves the estimate by a constant factor.

Theorem 3. *For an HU routine using $P_d^{\ell_1}$ in the diagonal update as defined in Eq.* (10)*, momentum as defined in Sec. 3.3 and a step length $\lambda = \frac{(1-\beta)^2}{2}\,\mathrm{tr}(\rho_H \Delta H)$, the maximum number of steps needed to find an ϵ-feasible solution to a feasible program* (5) *is upper bounded by*

$$T = 16(1-\beta)^{-6}\epsilon^{-2}\ln(n). \tag{32}$$

The proof is given in Appendix A.1.

Without the use of momentum terms (i.e. setting $\beta = 0$), this becomes $T = 16\epsilon^{-2}\ln(n)$, thus improving the bound of Ref. [25] by a factor of 4. We repeat that the numerically observed speedup of our improvements compared to Ref. [25] is much larger than the improvement of the *a priori* bounds.

We note that the proof of convergence no longer holds when using $P_d^{\ell_2}$ for the diagonal update as proposed in Sec. 3.2, while still using $P_d^{\ell_1}$ in the feasibility criteria, because the proof assumes $\mathrm{tr}(\rho_H\, P_d) \geq \epsilon$, which is not necessarily the case for mixed P_d's. Additionally, Thm. 3 does not consider an adaptive step size, as in Sec. 3.1. However, in practice, we find that these modifications do substantially improve algorithmic performance.

3.5 Numerical benchmarks of the non-asymptotic improvements

Here, we numerically study the effects of the different improvements made in Secs. 3.1-3.4.1. This section consists of three parts:

1. We observe the decrease in the number of iterations when applying the diagonal update $P_d^{\ell_2}$ instead of $P_d^{\ell_1}$ as described in Sec. 3.2, together with the momentum term described in Sec. 3.3 for different values of β.

2. We observe the decrease in the required number of iteration when successively applying all the improvements compared to the original algorithm.

3. We analyze how the number of iterations of the fully improved HU routine scales with ϵ.

The simulations are performed on 20 sparse QUBO instances of the block form given in Eq. (3). The sparsity pattern is chosen uniformly at random with sparsity $s = 16$. The non-zero elements are sampled from a standard Gaussian distribution, the matrices are normalized.

For the first part, we use instances with dimension $n = 1024$ and require a target accuracy $\epsilon = 0.001$. We run the complete HU routine using either $P_d^{\ell_1}$ or $P_d^{\ell_2}$ in all diagonal updates and test values for the momentum hyperparameter β between 0 and 0.7. To have a direct comparison, we choose to compare feasible instances of (2) with target objective values γ equal to the optimal objective value $\gamma^\star$, instead of applying full binary searches. The optimum $\gamma^\star$ is obtained by an SDP solver beforehand for each instance (specifically, the Splitting Conic Solver (SCS) described in Ref. [37]). The results are displayed in Fig. 3.

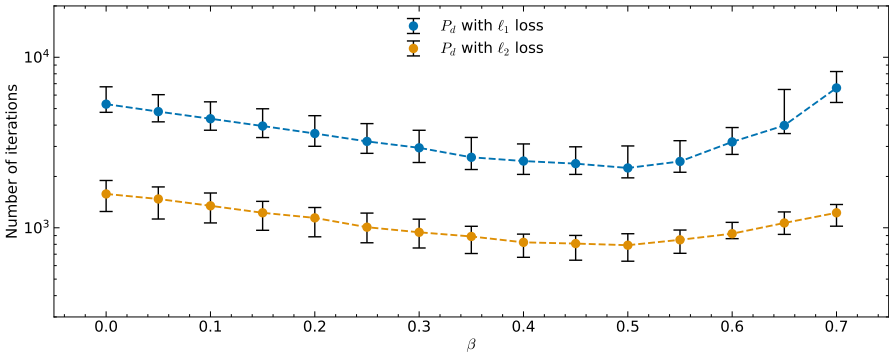

Figure 3: Number of iterations for varying values of the momentum hyperparameter β. The two different approaches for the diagonal update are compared: The original ℓ_1 norm based $P_d^{\ell_1}$ (blue) and the new ℓ_2 norm based $P_d^{\ell_2}$ (orange). The instances have dimension $n = 1024$, sparsity $s = 16$ and precision $\epsilon = 0.001$, and use the optimal SDP solution $\gamma^\star$ as the target objective value. The error bars show the upper and lower quartiles.

For the second part, to compare the improved algorithm to the original one, we evaluate three categories: (i) solving a feasible instance, (ii) proving infeasibility of an instance, and (iii) performing a complete binary search. The numerical simulations were performed on instances with dimension $n = 128$ and target accuracy $\epsilon = 0.01$. For the feasible instances we used an optimal target objective value $\gamma^\star$ obtained from an SDP solver (SCS), while infeasible instances used the candidate $\gamma = \gamma^\star + 0.02$. The comparison is made both in terms of the number of iterations and the number of matrix exponentiations required. The latter make up the vast majority of computational cost in each iteration, and thus provide a good measure for comparing the running time of the different approaches. The results are displayed in Table 2.

cumulative improvements	feasible instance		infeasible instance		binary search	
	iterations	matrix exp.	iterations	matrix exp.	iterations	matrix exp.
original algorithm	88292	88292	3.11e+06†	3.11e+06†	1.28e+07†	1.28e+07†
with adaptive step size	171	241	3.11e+06†	4.27e+06*	1.26e+07†	1.55e+07*
with entropy tracking	171	241	104	144	802	1116
with ℓ_2 norm based P_d	62	86	54	72	323	439
with momentum	42	59	38	50	219	296

Table 2: Comparison in terms of HU iterations and matrix exponential computations for the original HU algorithm and the new one with the improvements from Secs. 3.1-3.4.1 applied cumulatively. The number of matrix exponentials is equal to the sum of the number of iterations and the number of overshoots. The results are averaged over 20 instances with dimension $n = 128$, sparsity $s = 16$ and precision $\epsilon = 0.01$. Values computed (partially) analytically using the termination criterion of Ref. [25, Thm. 2.1] are marked with a dagger (†). Values marked with an asterisk ($*$) are extrapolated: The total number of overshoots is estimated by observing the average number of overshoots per iteration and multiplying this with the analytical number of iterations from the termination criterion. We find a speedup by a factor of more than 1400 for solving the SDP for an optimal candidate $\gamma^\star$, and a speedup of more than 43000 for the complete binary search.

Finally, in the third part, we analyze the scaling of the number of iterations of the improved HU algorithm as a function of the precision ϵ. We have used instances with dimension $n = 1024$ and ϵ values between 10^{-2} and $10^{-3.4}$. The indicated number of iterations includes a full binary search to find the optimal value up to the desired precision. A numerical powerlaw extrapolation shows that the number of iterations scales as $0.006\epsilon^{-2.02}$. The results are shown in Fig. 4.

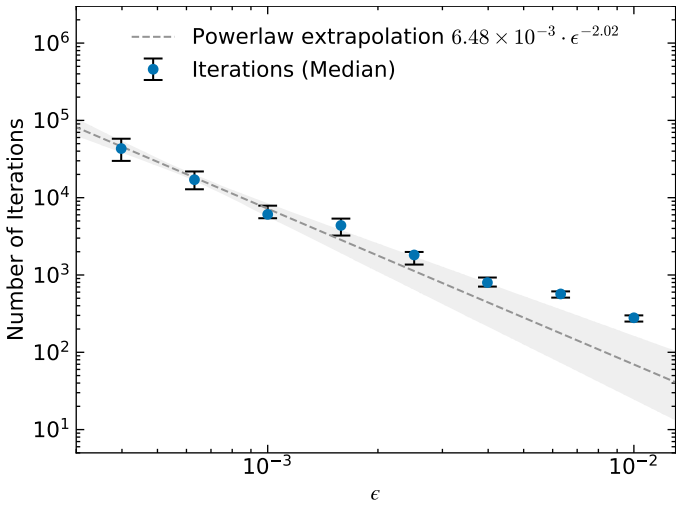

Figure 4: Total number of iterations over a complete binary search as a function of the precision ϵ. The fit is given by $f(\epsilon) = 6.48 \times 10^{-3}\epsilon^{-2.02}$ with a 95% confidence interval of $(-2.38, -1.73)$ for the exponent of ϵ. Therefore, the experimentally observed scaling exponent is similar to the theoretical bound of -2 (however, the experimentally observed prefactor is significantly better). The error bars show the upper and lower quartiles. The gray area shows the 95% confidence interval of the fit.

4 Improved randomized rounding

4.1 Analytical results

After solving the SDP relaxation, we still need to obtain a solution for the original QUBO problem (1). The Goemans-Williamson algorithm [26] provides a randomized rounding procedure for this purpose. One first computes the square root of ρ in terms of functional calculus, i.e. $\sqrt{\rho} \in \mathbb{R}^{n\times n}$ is a symmetric matrix with $\sqrt{\rho}\sqrt{\rho} = \rho$. Then, the entries of the rounded solution $x \in \{-1, 1\}^n$ are given by the sign of the column-wise

projection of $\sqrt{\rho}$ onto a Gaussian random vector:

$$\text{compute} \qquad \sqrt{\rho}, \tag{33}$$

$$\text{sample} \qquad g_j \overset{\text{iid}}{\sim} \mathcal{N}(0,1), \qquad j \in [n], \tag{34}$$

$$\text{compute} \qquad x_i \leftarrow \operatorname{sign}\left(\sum_j (\sqrt{\rho})_{ij} g_j\right), \qquad i \in [n]. \tag{35}$$

The Goemans-Williamson-style bounds on the quality of rounded solutions found in the literature are not directly applicable if the diagonal entries of ρ are only *approximately* equal to $1/n$. To address this issue, Ref. [25] proposes two different solutions: (1) First map the approximate optimizer ρ to a matrix $\rho^\sharp \in \mathbb{R}^{n\times n}$ that fulfills the diagonal constraints exactly, and then apply the above rounding procedure. (2) Apply the rounding procedure directly to the approximate optimizer ρ.

Both approaches show the same asymptotic scaling behavior.

Ref. [25] shows that if $\sum_i |\rho_{ii} - 1/n| \leq \epsilon$, the error of the final result scales as $\mathcal{O}(\epsilon^{1/4})$. Here, we adjust the parameters that go into the correction procedure, and improve the scaling to $\mathcal{O}(\epsilon^{1/3})$.

4.1.1 Correcting the SDP solution

The improved performance of the first approach, the one where the SDP solution is mapped to one which satisfies the constraints exactly, results from the theorem below.

Theorem 4. *There is an efficient procedure which, given an $\epsilon > 0$ and a psd matrix $\rho \in \mathbb{R}^{n\times n}$ such that $\sum_i |\rho_{ii} - 1/n| \leq \epsilon$, constructs a psd matrix $\rho^\sharp \in \mathbb{R}^{n\times n}$ where*

$$\rho^\sharp_{ii} = \frac{1}{n} \qquad \forall i \in [n] \tag{36}$$

$$\text{and} \qquad \left\|\rho^\sharp - \rho\right\|_{\text{tr}} = \mathcal{O}(\epsilon^{1/3}). \tag{37}$$

The proof is given in Appendix A.2.1.

Combining Thm. 4 with the matrix Hölder inequality allows us to bound the change in the objective value that results from applying this correction:

$$|\operatorname{tr}(C\rho^\sharp) - \operatorname{tr}(C\rho)| \leq \|C\| \left\|\rho^\sharp - \rho\right\|_{\text{tr}} = \mathcal{O}(\epsilon^{1/3}). \tag{38}$$

4.1.2 Rounding directly from the approximate solution

The improved results for the second approach – rounding directly – matches this scaling.

Theorem 5. *Let $\rho^\star \in \mathbb{R}^{n\times n}$ be the optimal SDP solution corresponding to a normalized cost matrix $C \in \mathbb{R}^{n\times n}$ with a block structure as defined in* (3)*. Let $\rho \in \mathbb{R}^{n\times n}$ be an approximate solution with*

$$\operatorname{tr}(C\rho^\star) - \operatorname{tr}(C\rho) \leq \epsilon, \tag{39}$$

$$\text{and} \qquad \sum_i |\rho_{ii} - \frac{1}{n}| \leq \epsilon \qquad \forall i \in [n], \tag{40}$$

with $0 \leq \epsilon \leq 1/2$. Let $x \in \{-1,1\}^n$ be the vector obtained by applying the randomized rounding procedure to ρ. Then

$$\mathbb{E}[x^T C x] \geq \left(\frac{4}{\pi} - 1\right) n \operatorname{tr}(C\rho^\star) - \mathcal{O}(n\epsilon^{1/3}). \tag{41}$$

The proof is given in Appendix A.2.2.

4.2 Numerical simulations for ϵ dependence

In the previous section, we have provided an analytic worst-case bound for the precision of the corrected SDP solution of $\mathcal{O}(\epsilon^{1/3})$. Now, we study the actual scaling behavior numerically. To this end, we use the matrices ρ that have been obtained as part of the numerical simulation described in Fig. 4, and apply the rounding procedure 10^5 times to each one. We quantify the performance of the rounding procedure in two ways: (1) The average objective value after rounding (because this is the quantity the theoretical guarantees make direct statements about). (2) The maximum objective value (because this is the number that would be used as the output of a numerical procedure). To reduce statistical fluctuations, the plots below show the average of 100 maximal values, computed for batches of size 1000 each.

In a second step, we compare the objective values obtained from rounding HU solutions to the objective values obtained from rounding exact solutions from an SDP solver (Splitting Conic Solver [37]). To this end, let $x_\epsilon, x_{\mathrm{opt}} \in \{-1,1\}^n$ be vectors obtained from applying randomized rounding to the HU solution and the SDP solver solution respectively. We then define the precision

$$\nu = (x_{\mathrm{opt}}^T C x_{\mathrm{opt}} - x_\epsilon^T C x_\epsilon)/n \tag{42}$$

of the rounded solution.

We display the results in Fig. 5 and summarize the estimated asymptotic scaling behaviors in Sec. 6.

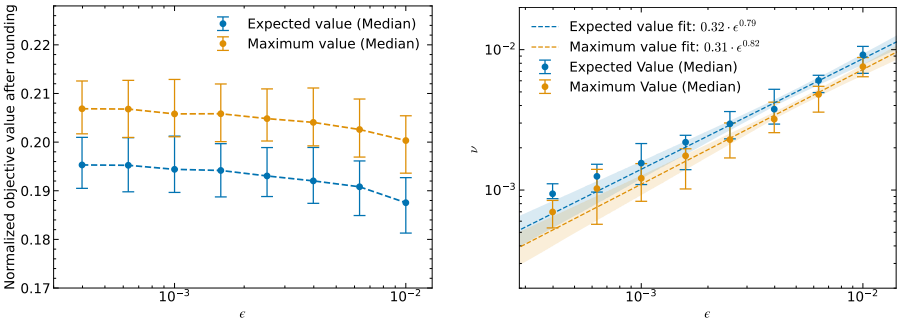

Figure 5: Behavior of the objective values $x^T Cx/n$ after rounding. Results are obtained by performing a binary search over HU instances for different precision parameters ϵ and applying randomized rounding. The results are averaged over 20 cost matrices with dimension $n = 1024$ and sparsity $s = 16$. The error bars show the upper and lower quartiles.
Left panel: Comparison of average objective value (blue) and the maximum value seen in 1000 roundings (orange).
Right panel: Difference $\nu = (x_{\text{opt}}^T Cx_{\text{opt}} - x_\epsilon^T Cx_\epsilon)/n$, where x_{opt} and x_ϵ have been obtained, respectively, by applying the rounding procedure to an optimal SDP solution, and an ϵ-precise HU solution. The fit for the average value is given by $f_\nu(\epsilon) = 0.32\epsilon^{0.79}$ with a 95% confidence interval of $(0.71, 0.88)$ for the exponent. The colored areas show the 95% confidence interval of the fits.

We observe that for the parameters tested, taking the best solution obtained from 1000 randomized rounding runs improves the final objective value significantly more than increasing the precision ϵ from 10^{-2} to $10^{-3.4}$. Because the rounding procedure is relatively computationally cheap, it seems advisable to work with a very high number of randomized roundings. We leave the task of determining the optimal tradeoff between the number of roundins and ϵ open for future work.

From the second part of the analysis, we find that the average of ν scales with $\mathcal{O}(\epsilon^{0.79})$. In contrast, the lower bounds for the two summands in Eq. (42) differ by a term of order $\mathcal{O}(\epsilon^{1/3})$. Therefore, the numerically observed scaling of ν as a function of ϵ is significantly better than a naive estimate based on the difference of the lower bounds would have suggested.

5 Improved Gibbs state simulation

In this section, we present improvements to the quantum version of the HU algorithm, which was originally given in Ref. [25]. On a high level, the idea of the quantum implementation is to realize the Gibbs state ρ as the *physical* state of a quantum system. The advantage of this method is that the dimension of matrices representable in this way scales exponentially with the number of qubits. The disadvantage is that information about the violation of the constraints and about the objective value have to be estimated statistically from physical measurements. We refer to Ref. [25] for a thorough description. In this section, we only account for the parts of the method for which we suggest improvements.

The quantum version of the HU algorithm delegates a number of subroutines to a quantum computer. More specifically, given a classical descriptions of H, P_c, and ΔH, it uses quantum subroutines to estimate the trace inner products $\mathrm{tr}(P_c\rho)$ and $\mathrm{tr}(\Delta H\rho)$, and the main diagonal elements ρ_{ii}. Recall that ρ is the Gibbs state for H.

As already stated in [25, Lem. 3.3], the quantum routine for estimating the diagonal elements dominates the running time. In their approach, obtaining an estimate for the probability distribution ρ_{ii} up to an error of $\mathcal{O}(\epsilon)$ in ℓ_1 norm for the Gibbs states ρ that appear in the HU algorithm requires a number of gates that scales as

$$\tilde{\mathcal{O}}(n^{3/2}s^{1/2+o(1)}\epsilon^{-5+o(1)}),$$

where the $\tilde{\mathcal{O}}$ notation hides logarithmic factors. In contrast, we will argue that this scaling can be improved to

$$\tilde{\mathcal{O}}(n^{3/2}s^{1/2+o(1)}\epsilon^{-3+o(1)}).$$

The improvement is mainly achieved by invoking newer and more optimal methods for working with Gibbs states on a quantum computer. Specifically, [25] was based on Ref. [39], while we switch to Ref. [5].

That reference, in turn, builds on a subroutine for *Hamiltonian ϵ-simulation*, i.e. for the task of implementing a unitary U that is ϵ-close to e^{itH} in operator norm, given access to an s-sparse matrix $H \in \mathbb{R}^{n\times n}$ stored in QRAM, and a time $t \in \mathbb{R}$.

The results of Ref. [5] are stated explicitly based on the Hamiltonian simulation algorithm of Ref. [12]. However, it is a straight-forward (if lengthy) exercise to swap in an improved method. For our analysis, we have done just that, using the routine from Ref. [33], which has complexity $\mathcal{O}(t\sqrt{s}\|H\|_{\ell_1\to\ell_2})^{1+o(1)}/\epsilon^{o(1)})$. Because $\|H\|_{\ell_1\to\ell_2} \leq \|H\|$, this allows to achieve a similar scaling in the sparsity as the original quantum HU routine from Ref. [25], while simultaneously having an improved dependence on the precision ϵ. The Hamiltonian Simulation subroutine enters the analysis of Ref. [5] in their Lemma 36. Plugging in the version of Ref. [33] and retracing the rest of the argument then gives the following performance for preparation of a Gibbs state:

Lemma 6 ([5, Lem. 44])**.** *Given QRAM access to an s-sparse matrix $H \in \mathbb{R}^{n\times n}$ satisfying $\mathbb{1} \prec H$ and $2\mathbb{1} \not\prec H$, we can probabilistically prepare a purified Gibbs state $|\tilde{\rho}\rangle_{AB}$, s.t. with high probability $\left\|\mathrm{Tr}_B(|\tilde{\rho}\rangle\langle\tilde{\rho}|_{AB}) - e^{-H}/\mathrm{tr}(e^{-H})\right\|_{\mathrm{tr}} \leq \frac{\epsilon}{8}$ holds, using*

$$\tilde{\mathcal{O}}((\|H\|\sqrt{s})^{1+o(1)}\sqrt{n})$$

queries and gates.

Lem. 6 requires H to satisfy satisfies $\mathbb{1} \prec H$ and $2\mathbb{1} \not\prec H$. We achieve this by employing a trick from Ref. [5, Cor. 14], where we compute an estimate $\tilde{\lambda}_{\min}$ of the minimum eigenvalue of H with an additive error $1/2$. Using this estimate, we apply Lem. 6 to a shifted Hamiltonian

$$H_+ := H - (\tilde{\lambda}_{\min} - 3/2)\mathbb{1} \tag{43}$$

that fulfills the requirements. Shifting a Hamiltonian by a multiple of the identity does not change the corresponding Gibbs state. Computing $\tilde{\lambda}_{\min}$ can be achieved using $\tilde{\mathcal{O}}(\|H\| s\sqrt{n})$ queries and gates (c.f. Ref. [5, Lem. 50] with $\epsilon = 1/2$).

As argued in Ref. [25, Sec. 3.4], for Hamiltonian matrices H that occur in the HU algorithm, one has the operator norm bound $\|H\| = \mathcal{O}(\log(n)\epsilon^{-1})$. Furthermore, they point out that, given $\mathcal{O}(n\epsilon^{-2})$ preparations of ρ with precision $\epsilon/8$, one can acquire estimate $\tilde{\rho}_{ii}$ for the diagonal entries fulfilling $\sum_i |\tilde{\rho}_{ii} - \rho_{ii}| = \epsilon/4$. Thus, the cost per iteration of the HU algorithm is

$$\tilde{\mathcal{O}}(n^{3/2} s^{1/2+o(1)} \epsilon^{-3+o(1)}),$$

as claimed.

6 Asymptotic performance of the improved Hamiltonian Updates

In this section, we summarize the improvement in the asymptotic performance of the HU procedure that we have achieved compared to Ref. [25].

The first table below gives estimates for the asymptotic complexity of solving the problem (8) as a function of ϵ. In this paper, we did not attempt to find improved rigorous asymptotic estimates for this scaling behavior. However, in Sec. 3, we have described several practical ways to speed up convergence. The numerical results presented in Sec. 3.5 show that these lead to speedups by very large constant factors, but the numerically observed asymptotic scaling matches the original exponent within the margin of error.

	Previous theor. bounds	New theor. bounds	Numerical scaling	95% CI for exponent
Number of iterations	$\tilde{\mathcal{O}}(\epsilon^{-2})$	(no new results)	$\tilde{\mathcal{O}}(\epsilon^{-2.02})$	$[-2.39, -1.74]$

The next table summarizes the scaling of the precision ν of the rounded solution (compared to an ideal SDP solution, see Eq. (42)) as a function of ϵ, as detailed in Sec. 4.1.

	Previous theor. bounds	New theor. bounds	Numerical scaling	95% CI for exponent
Precision after rounding	$\tilde{\mathcal{O}}(\epsilon^{0.25})$	$\tilde{\mathcal{O}}(\epsilon^{0.33})$	$\tilde{\mathcal{O}}(\epsilon^{0.79})$	$[0.71, 0.88]$

The improved scaling of the total running time of each iteration as a function of ϵ, as detailed in Sec. 5 is as follows

	Previous theor. bounds	New theor. bounds
Time per iteration:		
classical	$\tilde{\mathcal{O}}(\min\{n^3, n^2 s\epsilon^{-1}\})$	(no new results)
quantum	$\tilde{\mathcal{O}}(n^{1.5} s^{0.5+o(1)} \epsilon^{-5+o(1)})$	$\tilde{\mathcal{O}}(n^{1.5} s^{0.5+o(1)} \epsilon^{-3+o(1)})$

The previous tables can now be combined, to estimate the scaling of the total running time required to achieve a given precision ν of the solution after rounding. For

the final column, we have combined the analytic estimates on the running time from the previous table, with the numerically found scaling of the precision ν from the table above.

	Previous theor. bounds	New theor. bounds	Combined theor. / num. scaling
Total time			
classical	$\tilde{\mathcal{O}}(\min\{n^3\nu^{-8}, n^2s\nu^{-12}\})$	$\tilde{\mathcal{O}}(\min\{n^3\nu^{-6}, n^2s\nu^{-9}\})$	$\tilde{\mathcal{O}}(\min\{n^3\nu^{-2.56}, n^2s\nu^{-3.82}\})$
quantum	$\tilde{\mathcal{O}}(n^{1.5}s^{0.5+o(1)}\nu^{-28+o(1)})$	$\tilde{\mathcal{O}}(n^{1.5}s^{0.5+o(1)}\nu^{-15+o(1)})$	$\tilde{\mathcal{O}}(n^{1.5}s^{0.5+o(1)}\nu^{-6.35})$

Theorem 1 of [25] (reproduced in our introduction) stated their asymptotic performance in terms of the precision μ of the SDP solution, not in terms of the practically more relevant precision ν of after rounding. However, as we found in Sec. 4, the two quantities display the same scaling behavior as a function of ϵ. Hence, conversely, the running time scaling as a function of ν matches the running time scaling as a function of μ. Therefore, the results of the previous table are directly comparable with Theorem 1 of [25].

7 Non-asymptotic benchmarking of quantum implementations

In this section, we compare the quantum implementation of HU to its classical counterpart by estimating the required number of quantum gates and tracking the classical computation time. We then extrapolate these results to larger problem instances to explore for which problem sizes one can expect a future quantum computer to beat a classical one, under optimistic assumptions. This approach mirrors previous studies for the non-asymptotic behavior of quantum algorithms that are too large to be simulated in the gate model; see, e.g. [18, 42, 3, 40]. We assume that the reader is familiar with standard methods in quantum computing. For further background consult e.g. Ref. [35].

7.1 Gate counting

To compare the classical and quantum running times, we generate random problem instances and solve them using the classical version of the HU algorithm. We estimate the number of gates needed for the quantum subroutines and compare this with the classical running time. From this comparison, we determine the maximum allowable gate time for each instance that would enable a quantum computer to outperform its classical counterpart. For our estimates of the quantum gate count, we consistently make assumptions that are favorable to the quantum computer. This approach will be justified in retrospect: It will turn out that even these optimistic estimates put the threshold for a quantum advantage far beyond the capabilities of realistic hardware.

As discussed in Sec. 5, the running time of the quantum part is dominated by the task of estimating the diagonal elements ρ_{ii} of the Gibbs state. We will only estimate the cost of this part, and neglect all other quantum sub-routines. (Recall that this is justified, as we aim for an *optimistic* assessment of the quantum complexity). In the

benchmarks, the precision of the Gibbs state simulations is set to $\epsilon/8$, in line with Lem. 19 and Ref. [25, Lem. 3.3]. For simplicity, and again being generous to the quantum approach, we nevertheless assume that the subroutines return exact values. Additionally, we compute the free energy directly using Eq. (23) instead of bounding it via Eq. (31). Finally, we will count solely logical two-qubit gates, neglecting single-qubit gates and error-correction overheads.

As detailed in Appendix B, in this framework, we find the following result for the complexity of preparing Gibbs states.

Estimate 7. *Let $H \in \mathbb{R}^{n\times n}$ be an s-sparse matrix and b be the number of bits per entry used to store H. The expected number of two-qubit gates used to prepare an approximation of the Gibbs state ρ_H with precision ϵ using the constructions in Refs. [33, 20, 34, 5, 14] is at least*

$$(32b + 32\log_2(n) - 18)\,(4.5\ln(7.8\epsilon^{-1})\,n^{1/2}s\,\|H_+\|_{\max} - 1), \tag{44}$$

with H_+ constructed from H as defined in Eq. (43)*.*

The estimates for the diagonal entries of ρ are obtained statistically by measuring multiple prepared Gibbs states in the computational basis. The number of Gibbs state samples required can be bounded as follows:

Estimate 8. *Let $\epsilon \leq 1/4$. Let $\rho \in \mathbb{R}^{n\times n}$ be a quantum state that can be prepared with precision $\epsilon/8$ in trace distance on a quantum computer. The expected number of preparations needed to compute estimates $\hat{\rho}_{ii}$ for the diagonal entries of ρ, s.t. $\sum_i |\rho_{ii} - \hat{\rho}_{ii}| \leq \frac{\epsilon}{4}$ using the construction in Appendix B, is at least*

$$128\ln(2)\epsilon^{-2}n. \tag{45}$$

A detailed account for how to arrive at Estimates 7 and 8 is provided in Appendix B.

We compute the total number of two-qubit gates required for a diagonal update step in the HU routine by multiplying the gate cost per Gibbs state sample (44) with the expected number of samples needed to estimate the diagonal entries of ρ (45).

We avoid directly computing $\|H_+\|_{\max} = \|H - (\tilde{\lambda}_{\min} - 3/2)\mathbb{1}\|_{\max}$ in the benchmark, as determining the minimum eigenvalue classically for each iteration is computationally expensive. Instead, we use a conservative estimate $\|H_+\|_{\max} \gtrapprox \|H\|_{\max}$, based on our numerical observation $\|H_+\|_{\max} \approx 2\,\|H\|_{\max}$. Additionally, we assume that the matrix H is represented on the quantum architecture with just $b = 8$ logical bits per element.

We generate random cost matrices with a block form as defined in (3) with normally distributed non-zero entries in each block and a sparsity of $s = 16$. A total of 256 instances are created, with dimensions uniformly distributed between $512 \leq n \leq 4096$. We then solve the instances using the HU algorithm with $\epsilon = 0.01$ and a binary search with a final maximum gap of 0.01. The matrix exponentials – the most computationally expensive part of the process – are executed on an Nvidia GeForce RTX 4090 GPU card, while an Intel i7-13700 CPU handles the other operations in the routine.

7.2 Results

Here, we compare the quantum and the classical running times for the benchmarked non-asymptotic instances.

In a first step, we ignore the possibility of parallelizing the quantum implementation. With this in mind, we divide the time the classical implementation took by the lower bound on the number of quantum gates given in the last section. A necessary condition for a quantum architecture to be preferable to classical consumer hardware would be that its two-qubit gates run in time at most equal to that quotient.

The results are displayed in the left panel of Fig. 6. We find that even for large instances up to dimension $n = 4096$ a quantum computer requires a two-qubit gate time of less than 10^{-19}s to be able to break even. This is more than ten orders of magnitude away from the current single-qubit gate speed record of 6.5×10^{-9}s [19].

There are two parts of the quantum algorithm where parallelization can provide an advantage:

- *Parallel execution of quantum gate operations*. Gates acting on different qubits can be applied simultaneously rather than sequentially. The smallest number of layers required to execute a circuit taking this possibility into account is called its *depth*. Unfortunately, in our case, it is not known how to reduce the circuit depth significantly below the gate count. Indeed, the main building block of the circuit is an adder, for which the state-of-the-art two-qubit gate depth is only a factor of two lower than the total two-qubit gate count [20].

- *Parallel sampling of Gibbs states*. This step offers significant parallelization potential, as the required number of samples is more than $128 \ln(n) \epsilon^{-2} n$. From a computer science perspective, this reduction in algorithmic depth might be of theoretical interest. However, each simultaneous computation would require its own dedicated quantum computer, limiting the practical relevance.

7.2.1 Extrapolation

Due to the superior asymptotic scaling of the quantum HU algorithm compared to its classical counterpart, larger problem instances make it easier for the quantum approach to break even. We now investigate whether this favorable scaling is sufficient to achieve a quantum advantage for instances that remain computationally feasible at all. For this we use two different approaches: (1) Extrapolate using the exponents that have been derived theoretically in Sec. 6, and let the benchmark data determine just the prefactor. (2) Fit a powerlaw function of the form $f(t) = a_1 t^{a_2}$ with free parameters a_1, a_2 to the data.

Both approaches are displayed in Fig. 6. We have extrapolated the quantum gate counts for instance sizes that would take more than 100 years to be solved with the given classical hardware. Still, even at this scale, the required two-qubit gate time is more than a factor of 10^7 less than current quantum gate times. For a powerlaw extrapolation (2) the gap is even larger with a factor of more than 10^8.

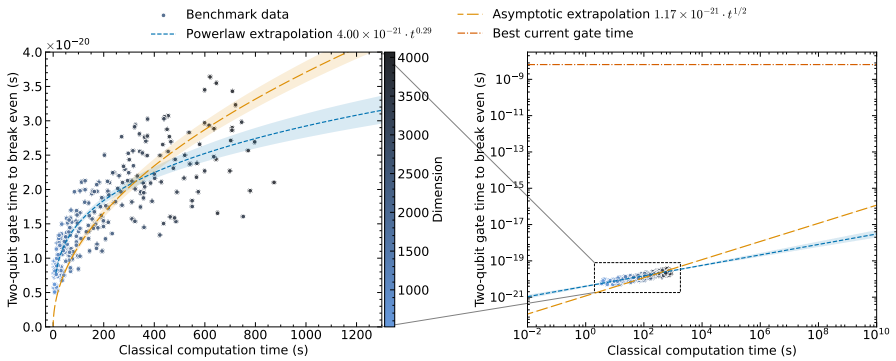

Figure 6: Allowed maximum two-qubit gate time for a quantum computer to break even to a classical simulation. The benchmark data is extrapolated based on the known asymptotic scaling behaviors (orange) and a power law fit (blue). The constant line (red) shows the best current classical single-qubit gate time of 6.5ns. The colored areas show the 95% confidence interval of the fits. We see that with this gate speed, even for running times of more than 100 years, the quantum implementation would be more than 10^7 times slower than the classical implementation.

8 Conclusion

In this paper, we have investigated whether the theoretically proven asymptotic speedups provided by quantum SDP solvers can be used to achieve a practical advantage for solving convex relaxations of combinatorial optimization problems. A priori, combinatorial problems seem to be a good fit for quantum SDP methods, because their main drawback – the unfavorable scaling in the precision – is less important for applications where the solution will be subjected to a rounding procedure.

Unfortunately, our work has found no indication that advantages manifest in regimes that are remotely realistic. This finding holds in spite of the fact that we have spent significant efforts to improve the Hamiltonian Updates algorithm before benchmarking it, and that we have made a large number of approximations and assumptions in favor of a quantum architecture (in particular, ignoring all quantum error correction overhead, and only estimating the cost of a subset of the routines a quantum computer would have to run).

It therefore seems that, to the best of our current knowledge, the proposed quantum SDP methods for combinatorial optimization may constitute a *galactic algorithm* – advantageous in theory, but such that their benefit requires instance sizes that are far beyond what can be realistically solved.

We note that the present work does not rigorously prove the absence of a realistic quantum advantage. Indeed, any such result would require rigorous and non-asymptotic lower bounds on the classical complexity of the problem. But unconditional complexity lower bounds are notoriously hard to obtain [7], so that the results of thorough benchmarks of state-of-the-art implementations seem to constitute the best evidence that is realistically achievable.

Data availability - The data that supports the findings of this article is openly available [29].

8.1 Acknowledgements

We thank Brandon Augustino, Johannes Berg, Lionel Dmello, and Sebastian Stiller for insightful discussions.

This work was supported by the Federal Ministry for Economic Affairs and Climate Action (BMWK), project ProvideQ, and the German Federal Ministry of Education and Research (BMBF), project QuBRA. FH and DG are also supported by Germany's Excellence Strategy – Cluster of Excellence Matter and Light for Quantum Computing (ML4Q) EXC 2004/1 (390534769).

References

[1] Amira Abbas et al. "Challenges and opportunities in quantum optimization". In: *Nature Reviews Physics* (2024). DOI: 10.1038/s42254-024-00770-9.

[2] Noga Alon and Assaf Naor. "Approximating the Cut-Norm via Grothendieck's Inequality". In: *SIAM Journal on Computing* (2006). DOI: 10.1137/S0097539704441629.

[3] Sabrina Ammann et al. *Realistic Runtime Analysis for Quantum Simplex Computation*. 2023. eprint: 2311.09995.

[4] Joran van Apeldoorn and András Gilyén. "Improvements in Quantum SDP-Solving with Applications". In: *46th International Colloquium on Automata, Languages, and Programming (ICALP 2019)*. 2019. DOI: 10.4230/LIPIcs.ICALP.2019.99.

[5] Joran van Apeldoorn et al. "Quantum SDP-Solvers: Better upper and lower bounds". In: *Quantum* (2020). DOI: 10.22331/q-2020-02-14-230.

[6] Huzihiro Araki and Elliott H. Lieb. "Entropy inequalities". English (US). In: *Communications In Mathematical Physics* (1970). DOI: 10.1007/BF01646092.

[7] S. Arora and B. Barak. *Computational Complexity: A Modern Approach*. Cambridge University Press, 2009.

[8] S. Arora, E. Hazan, and S. Kale. "Fast algorithms for approximate semidefinite programming using the multiplicative weights update method". In: *46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*. 2005. DOI: 10.1109/SFCS.2005.35.

[9] Brandon Augustino. *Private communication*. 2024.

[10] Brandon Augustino et al. *Solving the semidefinite relaxation of QUBOs in matrix multiplication time, and faster with a quantum computer*. 2023. arXiv: 2301.04237 [quant-ph].

[11] Alexander Barvinok. *A course in convexity*. American Mathematical Soc., 2002.

[12] Dominic W. Berry, Andrew M. Childs, and Robin Kothari. “Hamiltonian Simulation with Nearly Optimal Dependence on all Parameters”. In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. DOI: 10.1109/focs.2015.54.

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

[14] Michel Boyer et al. “Tight bounds on quantum searching”. In: *Fortschritte der Physik: Progress of Physics* (1998).

[15] Fernando G. S. L. Brandao et al. *Quantum SDP Solvers: Large Speed-ups, Optimality, and Applications to Quantum Learning*. 2019. arXiv: 1710.02581 [quant-ph].

[16] Fernando G.S.L. Brandao and Krysta M. Svore. “Quantum Speed-Ups for Solving Semidefinite Programs”. In: *2017 IEEE 58th Annual Symposium on Foundations of Computer Science (FOCS)*. 2017. DOI: 10.1109/FOCS.2017.45.

[17] Jop Briet, Fernando Mario de Oliveira Filho, and Frank Vallentin. “Grothendieck inequalities for semidefinite programs with rank constraint”. In: *Theory of Computing* (2014). DOI: 10.4086/toc.2014.v010a004.

[18] Chris Cade et al. “Quantifying Grover speed-ups beyond asymptotic analysis”. In: *Quantum* (2023). DOI: 10.22331/q-2023-10-10-1133.

[19] Y. Chew et al. “Ultrafast energy exchange between two single Rydberg atoms on a nanosecond timescale”. In: *Nature Photonics* (2022). DOI: 10.1038/s41566-022-01047-2.

[20] Steven A. Cuccaro et al. *A new quantum ripple-carry addition circuit*. 2004. arXiv: quant-ph/0410184 [quant-ph].

[21] Alexander M. Dalzell et al. “End-To-End Resource Analysis for Quantum Interior-Point Methods and Portfolio Optimization”. In: *PRX Quantum* (2023). DOI: 10.1103/prxquantum.4.040325.

[22] Alexander M. Dalzell et al. *Quantum algorithms: A survey of applications and end-to-end complexities*. 2023. arXiv: 2310.03011 [quant-ph].

[23] R.P. Feynman. *Statistical Mechanics: A Set Of Lectures*. Advanced Books Classics. Avalon Publishing, 1998.

[24] Shmuel Friedland and Lek-Heng Lim. *Symmetric Grothendieck inequality*. 2020. arXiv: 2003.07345 [math.FA].

[25] Fernando G.S L. Brandão, Richard Kueng, and Daniel Stilck França. “Faster quantum and classical SDP approximations for quadratic binary optimization”. In: *Quantum* (2022). DOI: 10.22331/q-2022-01-20-625.

[26] 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*. 1994. DOI: 10.1145/195058.195216.

[27] Lov K. Grover. "A Fast Quantum Mechanical Algorithm for Database Search". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on the Theory of Computing*. ACM, 1996. DOI: 10.1145/237814.237866.

[28] Aram W. Harrow, Avinatan Hassidim, and Seth Lloyd. "Quantum Algorithm for Linear Systems of Equations". In: *Phys. Rev. Lett.* (15 2009). DOI: 10.1103/PhysRevLett.103.150502.

[29] Fabian Henze et al. 2025. DOI: 10.5281/zenodo.14871936.

[30] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. 1972. DOI: 10.1007/978-1-4684-2001-2_9.

[31] Christopher King. *Inequalities for Trace Norms of 2 × 2 Block Matrices*. 2003. DOI: 10.1007/s00220-003-0955-9.

[32] Yin Tat Lee, Aaron Sidford, and Sam Chiu-Wai Wong. "A Faster Cutting Plane Method and its Implications for Combinatorial and Convex Optimization". In: *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*. 2015. DOI: 10.1109/FOCS.2015.68.

[33] Guang Hao Low. "Hamiltonian simulation with nearly optimal dependence on spectral norm". In: *Proceedings of the 51st Annual ACM SIGACT Symposium on Theory of Computing*. STOC 2019. Phoenix, AZ, USA: Association for Computing Machinery, 2019. DOI: 10.1145/3313276.3316386.

[34] Guang Hao Low and Isaac L. Chuang. "Hamiltonian Simulation by Qubitization". In: *Quantum* (2019). DOI: 10.22331/q-2019-07-12-163.

[35] Michael A. Nielsen and Isaac L. Chuang. *Quantum Computation and Quantum Information: 10th Anniversary Edition*. Cambridge University Press, 2010.

[36] Ryan O'Donnell. "SOS Is Not Obviously Automatizable, Even Approximately". In: *8th Innovations in Theoretical Computer Science Conference*. 2017.

[37] Brendan O'Donoghue et al. "Conic Optimization via Operator Splitting and Homogeneous Self-Dual Embedding". In: *Journal of Optimization Theory and Applications* (2016). DOI: 10.1007/s10957-016-0892-3.

[38] Boris T. Polyak. "Some methods of speeding up the convergence of iteration methods". In: *USSR Computational Mathematics and Mathematical Physics* (1964). DOI: https://doi.org/10.1016/0041-5553(64)90137-5.

[39] David Poulin and Pawel Wocjan. "Sampling from the Thermal Quantum Gibbs State and Evaluating Partition Functions with a Quantum Computer". In: *Phys. Rev. Lett.* (22 2009). DOI: 10.1103/PhysRevLett.103.220502.

[40] *QuBRA Quantum Benchmarking Project*. https://github.com/qubrabench. 2024.

[41] Prasad Raghavendra and Benjamin Weitz. "On the Bit Complexity of Sum-of-Squares Proofs". In: *44th International Colloquium on Automata, Languages, and Programming*. 2017.

[42] Debora Ramacciotti, Andreea I. Lefterovici, and Antonio F. Rotundo. “Simple quantum algorithm to efficiently prepare sparse states”. In: *Phys. Rev. A* (3 2024). DOI: 10.1103/PhysRevA.110.032609.

[43] Joseph Renes. *Quantum Information Theory: Concepts and Methods*. Walter de Gruyter GmbH & Co KG, 2022.

[44] Vivek V. Shende and Igor L. Markov. “On the CNOT-cost of TOFFOLI gates”. In: *Quantum Info. Comput.* (2009).

[45] Peter W. Shor. “Algorithms for Quantum Computation: Discrete Logarithms and Factoring”. In: *35th Annual Symposium on Foundations of Computer Science*. IEEE Computer Society, 1994. DOI: 10.1109/SFCS.1994.365700.

A Proofs

A.1 Free energy tracking

Lem. 2 describes a well-known concept in statistical mechanics [23, Ch. 2.11]. We still give the calculation for completeness.

Lemma 2. *For all symmetric $H, \Delta H \in \mathbb{R}^{n\times n}$ and $\lambda \in \mathbb{R}$, the free energy satisfies*

$$\partial_\lambda F(H + \lambda\Delta H) = \operatorname{tr}(\rho_{H+\lambda\Delta H}\lambda\Delta H). \tag{29}$$

Proof. We have

$$\begin{aligned}
\partial_\lambda \operatorname{tr}(e^{-H-\lambda\Delta H}) &= \sum_{k=0}^{\infty} \frac{1}{k!} \operatorname{tr}\left(\partial_\lambda(-H-\lambda\Delta H)^k\right) \\
&= \sum_{k=1}^{\infty} \frac{-1}{k!} \sum_{i=0}^{k-1} \operatorname{tr}\left((-H-\lambda\Delta H)^i \Delta H (-H-\lambda\Delta H)^{k-i-1}\right) \\
&= \sum_{k=1}^{\infty} \frac{-1}{(k-1)!} \operatorname{tr}\left((-H-\lambda\Delta H)^{k-1}\Delta H\right) \\
&= -\operatorname{tr}\left(e^{-H-\lambda\Delta H}\Delta H\right),
\end{aligned} \tag{46}$$

where the key step was to use the cyclicity of the trace to combine the terms $(-H - \lambda\Delta H)$. Then

$$\begin{aligned}
\partial_\lambda F(H+\lambda\Delta H) &= -\partial_\lambda \ln \operatorname{tr}(e^{-H-\lambda\Delta H}) \\
&= -\frac{\partial_\lambda \operatorname{tr}(e^{-H-\lambda\Delta H})}{\operatorname{tr}(e^{-H-\lambda\Delta H})} \\
&= \frac{\operatorname{tr}(e^{-H-\lambda\Delta H}\Delta H)}{\operatorname{tr}(e^{-H-\lambda\Delta H})} \\
&= \operatorname{tr}(\rho_{H+\lambda\Delta H}\lambda\Delta H).
\end{aligned} \tag{47}$$

□

Next, we prove Thm. 3. In preparation for this, we need the following Lemmas 9 and 10.

Lemma 9. *Let $0 \leq \beta < 1$ be the momentum hyperparameter, $\Delta H \in \mathbb{R}^{n\times n}$ be an update matrix in the HU routine and*

$$\lambda = \frac{(1-\beta)^2}{2} \operatorname{tr}(\rho_H \Delta H) \tag{48}$$

the step length of an update. Then,

$$\inf_{c\in\mathbb{R}} \|\Delta H - c\mathbb{1}\| \leq \frac{1}{1-\beta}. \tag{49}$$

Proof. Let K be the current iteration of HU and $(\Delta H)^{(k)}$, $(P)^{(k)}$ the operators corresponding to the k^{th} iteration. The chosen step length λ is independent of the update type (i.e. cost or diagonal update). Then, following from the definitions of the momentum in Sec. 3.3, the update term takes the form

$$(\Delta H)^{(K)} = \sum_{k=0}^{K-1} \beta^k (P)^{(K-k)}, \tag{50}$$

where, depending on the type of update in the respective iteration, $(P)^{(K-k)}$ takes the form of one of the following:

$$\begin{aligned} P_c &= \gamma \mathbb{1} - C \\ (P_d^{\ell_1})^{(K-k)} &= \operatorname{sign}\left((\operatorname{diag}\left(\rho^{(K-k)}\right) - \mathbb{1}/n \right) \\ (P_d^{\ell_2})^{(K-k)} &= \frac{\operatorname{diag}\left(\rho^{(K-k)}\right) - \mathbb{1}/n}{\|(\operatorname{diag}\left(\rho^{(K-k)}\right) - \mathbb{1}/n\|} \end{aligned} \tag{51}$$

In either case, this fulfills

$$\inf_{c\in\mathbb{R}} \left\| (P)^{(K-k)} - c\mathbb{1} \right\| \leq 1. \tag{52}$$

Then, from a geometric series argument follows

$$\begin{aligned} &\inf_{c\in\mathbb{R}} \left\| (\Delta H)^{(K)} - c\mathbb{1} \right\| \\ &\leq \sum_{k=0}^{K-1} \beta^k \inf_{c\in\mathbb{R}} \left\| (P)^{(K-k)} - c\mathbb{1} \right\| \\ &\leq \sum_{k=0}^{K-1} \beta^k \leq \frac{1}{1-\beta}. \end{aligned} \tag{53}$$

□

Next, we show that for a suitable step length the change in free energy (and thus also the decrease in relative entropy distance) can be lower-bounded in terms of $\operatorname{tr}(\rho_H \Delta H)$:

Lemma 10. *Let $0 \leq \beta < 1$ be the momentum hyperparameter, $\Delta H \in \mathbb{R}^{n\times n}$ an update matrix in the HU routine, ρ_H a Gibbs state for a Hamiltonian $H \in \mathbb{R}^{n\times n}$, $F(H) = -\ln(\operatorname{tr}(\exp(-H)))$ the free energy and*

$$\lambda = \frac{(1-\beta)^2}{2} \operatorname{tr}(\rho_H \Delta H) \tag{54}$$

the step length of an update. If $\operatorname{tr}(\rho_H \Delta H) \geq 0$, then

$$F(H + \lambda \Delta H) - F(H) \geq \frac{\lambda^2 (1-\beta)^2}{4}. \tag{55}$$

Proof. We denote $J := 1/(1-\beta)$. The first step is to show that

$$|\partial_\lambda^2 F(H+\lambda\Delta H)| = |\partial_\lambda \operatorname{tr}(\rho_{H+\lambda\Delta H}\Delta H)| \leq 2J^2. \tag{56}$$

First, note that Gibbs states are unchanged when adding multiples of identity to the Hamiltonian, so that

$$\rho_{H+\lambda\Delta H} = \frac{e^{-H-\lambda(\Delta H - c\mathbb{1})}}{\operatorname{tr}\left(e^{-H-\lambda(\Delta H-c\mathbb{1})}\right)}, \tag{57}$$

and similarly

$$\operatorname{tr}\left((\rho_{H+\lambda\Delta H} - \rho_{H+(\lambda+\varepsilon)\Delta H})c\mathbb{1}\right) = 0, \tag{58}$$

where, as shown in Lem. 9, we can choose c such that $\|\Delta H - c\mathbb{1}\| \leq J$.
Next, Ref. [16, Lem. 16] states for Hermitian operators H and H'

$$\left\| \frac{e^H}{\operatorname{tr}(e^H)} - \frac{e^{H'}}{\operatorname{tr}(e^{H'})} \right\| \leq 2\left(e^{\|H-H'\|} - 1\right). \tag{59}$$

Then, combining the above with a matrix Hölder inequality gives

$$\begin{aligned}
|\partial_\lambda \operatorname{tr}\left(\rho_{H+\lambda\Delta H}\Delta H\right)| &= \lim_{\varepsilon\to 0}\left|\frac{\operatorname{tr}\left(\rho_{H+\lambda\Delta H}\Delta H\right) - \operatorname{tr}\left(\rho_{H+(\lambda+\varepsilon)\Delta H}\Delta H\right)}{\varepsilon}\right| \\
&= \lim_{\varepsilon\to 0}\frac{\inf_{c\in\mathbb{R}}\left|\operatorname{tr}\left((\rho_{H+\lambda\Delta H} - \rho_{H+(\lambda+\varepsilon)\Delta H})(\Delta H - c\mathbb{1})\right)\right|}{|\varepsilon|} \\
&\leq \inf_{c\in\mathbb{R}}\|\Delta H - c\mathbb{1}\| \lim_{\varepsilon\to 0}\frac{\|\rho_{H+\lambda\Delta H} - \rho_{H+(\lambda+\varepsilon)\Delta H}\|_{\mathrm{tr}}}{|\varepsilon|} \\
&\leq 2J \lim_{\varepsilon\to 0}\frac{e^{\inf_{c\in\mathbb{R}}\|\varepsilon(\Delta H - c\mathbb{1})\|} - 1}{|\varepsilon|} \\
&= 2J \inf_{c\in\mathbb{R}}\|(\Delta H - c\mathbb{1})\| \leq 2J^2
\end{aligned} \tag{60}$$

and thus $|\partial_\lambda \operatorname{tr}(\rho_{H+\lambda\Delta H}\Delta H)| \leq 2J^2$. It follows that $\lambda = \operatorname{tr}(\rho_H \Delta H)/(2J^2)$ fulfills $\operatorname{tr}(\rho_{H+\lambda\Delta H}\Delta H) \geq 0$.

Next, let $\alpha := \operatorname{tr}(\rho_H \Delta H)$. Now, for the change in free energy we have

$$\begin{aligned}
\Delta F &= F\left(H + \alpha/(2J^2)\Delta H\right) - F(H) \\
&= \int_0^{\alpha/(2J^2)} \partial_{\lambda'} F(H) d\lambda' = \int_0^{\alpha/(2J^2)} \operatorname{tr}(\rho_{H+\lambda'\Delta H}\Delta H) d\lambda' \\
&\geq \int_0^{\alpha/(2J^2)} \left(\operatorname{tr}(\rho_H \Delta H) - \lambda' |\partial_{\lambda'} \operatorname{tr}(\rho_{H+\lambda'\Delta H}\Delta H)|\right) d\lambda',
\end{aligned} \tag{61}$$

where we bounded the decrease of the integrand by its derivative. Then, with (56) follows

$$\Delta F \geq \int_0^{\alpha/(2J^2)} (\alpha - 2J^2\lambda')d\lambda' = \frac{\alpha^2}{4J^2}. \tag{62}$$

□

Now, we have all the tools to bound the maximum number of iterations needed in the HU routine if a feasible solution exists:

Theorem 3. *For an HU routine using $P_d^{\ell_1}$ in the diagonal update as defined in Eq.* (10)*, momentum as defined in Sec. 3.3 and a step length $\lambda = \frac{(1-\beta)^2}{2}\operatorname{tr}(\rho_H \Delta H)$, the maximum number of steps needed to find an ϵ-feasible solution to a feasible program* (5) *is upper bounded by*

$$T = 16(1-\beta)^{-6}\epsilon^{-2}\ln(n). \tag{32}$$

Proof. By the definition in Sec. 3.3, ΔH is of the form $P + cM$ with some positive factor c (we can ignore here the rescaling $\tilde{P}_c = \operatorname{tr}(P_c\rho)P_c$, as this can be absorbed by the λ). Due to the overshoot criterion in each preceding iteration, $\operatorname{tr}(\rho_H M)$ is always nonnegative, and thus, $\operatorname{tr}(\rho_H \Delta H) \geq \operatorname{tr}(\rho_H P)$. With Lem. 10 we can lower bound the change in free energy at each step by $\Delta F \geq \frac{\operatorname{tr}(\rho_H \Delta H)^2(1-\beta)^6}{16}$. Then, when using the original $P_d^{\ell_1}$ for diagonal updates, we have $\operatorname{tr}(\rho_H P) \geq \epsilon$ at each update, and thus $\Delta F \geq \frac{\epsilon^2(1-\beta)^6}{16}$ in each iteration. The initial free energy is $F(0) = -\ln(n)$. If a feasible solution exists, the free energy cannot become positive, hence, the maximum number of steps is $T = 16(1-\beta)^{-6}\epsilon^{-2}\ln(n)$. □

A.2 Randomized rounding

A.2.1 Inexact diagonal constraints

The following proof of Thm. 4 is based on the proof of Ref. [25, Prop. 3.1] with the main difference being Eq. (74), which allows us to obtain a scaling of $\mathcal{O}(\epsilon^{1/3})$ instead of $\mathcal{O}(\epsilon^{1/4})$. For completeness we will give the full proof.

Theorem 4. *There is an efficient procedure which, given an $\epsilon > 0$ and a psd matrix $\rho \in \mathbb{R}^{n\times n}$ such that $\sum_i |\rho_{ii} - 1/n| \leq \epsilon$, constructs a psd matrix $\rho^\sharp \in \mathbb{R}^{n\times n}$ where*

$$\rho^\sharp_{ii} = \frac{1}{n} \qquad \forall i \in [n] \tag{36}$$

$$\text{and} \qquad \left\|\rho^\sharp - \rho\right\|_{\mathrm{tr}} = \mathcal{O}(\epsilon^{1/3}). \tag{37}$$

Proof. For convenience we define $\xi := \epsilon^{1/3}$. Then, by assumption,

$$\sum_i |\rho_{ii} - 1/n| \leq \xi^3. \tag{63}$$

The construction of $\rho^\sharp$ consists of two steps. First we adjust the rows and columns of ρ that have large diagonal deviations. For this, we set the diagonal elements in these rows and columns to $1/n$ while setting the corresponding off-diagonal entries to zero, thus ensuring positive semidefiniteness of the resulting matrix. This is a rather harsh correction, but $\sum_i |\rho_{ii} - 1/n| \leq \xi^3$ guarantees us that only a small number of entries need to be treated this way. In the second step, we set all remaining diagonal entries to $1/n$ and restore positive semidefiniteness by shifting ρ by $\frac{\xi}{n}\mathbb{1}$ and renormalizing it.

Denote $d_i = \rho_{ii} - 1/n$. Next, we define the index set $B \subset \{1, \ldots, n\}$ corresponding to the diagonals with larger deviations:

$$B = \{i : |d_i| > \frac{\xi}{n}\} \quad \text{with complement} \quad \bar{B} = \{i : |d_i| \leq \frac{\xi}{n}\}. \tag{64}$$

Now, we define the two matrices $\rho', D \in \mathbb{R}^{n\times n}$ and construct $\rho^\sharp$ from these, given as follows:

Step 1:

$$\rho'_{ij} = \begin{cases} 1/n & \text{if} \quad i = j, \quad i \in B, \\ 0 & \text{if} \quad i \neq j, \quad i \in B \vee j \in B, \\ \rho_{ij} & \text{else}, \end{cases} \tag{65}$$

Step 2:

$$D = \begin{cases} -d_i & \text{if} \quad i = j, \quad i \notin B, \\ 0 & \text{else}, \end{cases} \tag{66}$$

$$\rho^\sharp = \frac{1}{1+\xi}\left(\rho' + D + \frac{\xi}{n}\mathbb{1}\right). \tag{67}$$

Then ρ' is psd, because it arises by first compressing ρ to the submatrix with indices in $\bar{B}$, and then adding $\mathbb{1}/n$ to submatrix with indices in B. Both steps preserve positive-semidefiniteness. Next, D is a diagonal matrix with $|D_{ii}| \leq \frac{\xi}{n}$, and therefore, $D + \frac{\xi}{n}\mathbb{1}$ is psd. Thus, $\rho^\sharp$ is psd with diagonal entries $1/n$. We now show that these corrections are mild in the sense that $\left\|\rho^\sharp - \rho\right\|_{\mathrm{tr}} = \mathcal{O}(\xi)$.

From $\sum_i |\rho_{ii} - 1/n| \leq \xi^3$ we have

$$|B| \leq \xi^2 n. \tag{68}$$

Next, write ρ in block matrix form

$$\rho = \begin{pmatrix} \rho^{(11)} & \rho^{(12)} \\ \rho^{(21)} & \rho^{(22)} \end{pmatrix}, \tag{69}$$

with coordinates ordered such that the first block corresponds to the indices in B and the second block to the ones in $\bar{B}$. Then, following from the construction in (65),

$$\begin{aligned} \|\rho' - \rho\|_{\mathrm{tr}} &= \left\| \begin{pmatrix} \mathbb{1}_B/n - \rho^{(11)} & -\rho^{(12)} \\ -\rho^{(21)} & 0 \end{pmatrix} \right\|_{\mathrm{tr}} \\ &\leq \|\rho^{(11)}\|_{\mathrm{tr}} + 2\|\rho^{(12)}\|_{\mathrm{tr}} + \|\mathbb{1}_B/n\|_{\mathrm{tr}}. \end{aligned} \tag{70}$$

Now, using a result from Ref. [31], we have

$$\left\| \begin{pmatrix} \|\rho^{(11)}\|_{\mathrm{tr}} & \|\rho^{(12)}\|_{\mathrm{tr}} \\ \|\rho^{(21)}\|_{\mathrm{tr}} & \|\rho^{(22)}\|_{\mathrm{tr}} \end{pmatrix} \right\|_2 \leq \left\| \begin{pmatrix} \|\rho^{(11)}\|_{\mathrm{tr}} & \|\rho^{(12)}\|_{\mathrm{tr}} \\ \|\rho^{(21)}\|_{\mathrm{tr}} & \|\rho^{(22)}\|_{\mathrm{tr}} \end{pmatrix} \right\|_{\mathrm{tr}} \tag{71}$$

$$\leq \left\| \begin{pmatrix} \rho^{(11)} & \rho^{(12)} \\ \rho^{(21)} & \rho^{(22)} \end{pmatrix} \right\|_{\mathrm{tr}} = 1, \tag{72}$$

where $\|\cdot\|_2$ is the Frobenius (or Schatten-2) norm. Then,

$$\left\|\rho^{(11)}\right\|_{\mathrm{tr}}^2 + 2\left\|\rho^{(12)}\right\|_{\mathrm{tr}}^2 + \left\|\rho^{(22)}\right\|_{\mathrm{tr}}^2 \leq 1. \tag{73}$$

Because $\rho^{(22)}$ is a principle submatrix of ρ, it is also positive semidefinite. Thus,

$$\begin{aligned} \left\|\rho^{(22)}\right\|_{\mathrm{tr}} &= \mathrm{tr}\left(\rho^{(22)}\right) = 1 - \mathrm{tr}\left(\rho^{(11)}\right) \geq 1 - \frac{|B|}{n} - \sum_{i\in B} |d_i| \geq 1 - \xi^2 - \xi^3 \\ &= 1 - \mathcal{O}(\xi^2). \end{aligned} \tag{74}$$

Then, combining (73) and (74) gives $\left\|\rho^{(11)}\right\|_{\mathrm{tr}}^2 + 2\left\|\rho^{(12)}\right\|_{\mathrm{tr}}^2 = O(\xi^2)$, and hence

$$\left\|\rho^{(11)}\right\|_{\mathrm{tr}} + 2\left\|\rho^{(12)}\right\|_{\mathrm{tr}} = \mathcal{O}(\xi). \tag{75}$$

Furthermore, we have $\|\mathbb{1}_B/n\|_{\mathrm{tr}} = \frac{|B|}{n} \leq \xi^2$. Then, from (70),

$$\|\rho' - \rho\|_{\mathrm{tr}} = \mathcal{O}(\xi). \tag{76}$$

This concludes the first step. For the second step we have

$$\begin{aligned} \left\|\rho^\sharp - \rho'\right\|_{\mathrm{tr}} &= \left\| \left(\frac{1}{1+\xi} - 1\right)\rho' + \frac{1}{1+\xi}\left(D + \frac{\xi}{n}\mathbb{1}\right) \right\|_{\mathrm{tr}} \\ &= \frac{1}{1+\xi}\left\| -\xi\rho' + D + \frac{\xi}{n}\mathbb{1} \right\|_{\mathrm{tr}} \\ &\leq \frac{1}{1+\xi}\left(\xi\left\|\rho' - \rho\right\|_{\mathrm{tr}} + \xi\left\|\rho\right\|_{\mathrm{tr}} + \left\|D\right\|_{\mathrm{tr}} + \frac{\xi}{n}\left\|\mathbb{1}\right\|_{\mathrm{tr}}\right) \\ &= \mathcal{O}(\xi). \end{aligned} \tag{77}$$

Combining (76) and (77) with a triangle inequality gives us

$$\left\|\rho^\sharp - \rho\right\|_{\mathrm{tr}} = \mathcal{O}(\xi) = \mathcal{O}(\epsilon^{1/3}). \tag{78}$$

□

A.2.2 Approximation ratio after rounding

The proof of Lem. 11 closely follows Ref. [2, Sec. 4.1]. Compared to the original version of the proof, we also make a statement about the quality of rounded vectors obtained from solutions that are not strictly feasible.

Lemma 11. *Let $\rho^\star \in \mathbb{R}^{n\times n}$ be the optimal SDP solution corresponding to a cost matrix $C \in \mathbb{R}^{n\times n}$ with a block structure as defined in* (3)*. Let $x \in \{-1,1\}^n$ be the vector obtained by applying the randomized rounding procedure to a psd matrix $\rho \in \mathbb{R}^{n\times n}$ with $\rho_{ii} > 0$. Let $\sigma \in \mathbb{R}^{n\times n}$ be the matrix with entries $\sigma_{ij} = \frac{\rho_{ij}}{n\sqrt{\rho_{ii}\rho_{jj}}}$. Then*

$$\mathbb{E}\left[x^T C x\right] \geq \frac{2}{\pi} n \operatorname{tr}(C\sigma) - \left(1 - \frac{2}{\pi}\right) n \operatorname{tr}(C\rho^\star). \tag{79}$$

Proof. Set $u_i := \frac{(\sqrt{\rho})_i}{\sqrt{\rho_{ii}}}$, so that

$$u_i^T u_j = n\sigma_{ij}. \tag{80}$$

Then we have

$$x_i := \operatorname{sign}((\sqrt{\rho})_i^T g) = \operatorname{sign}(\sqrt{\rho_{ii}} u_i^T g) = \operatorname{sign}(u_i^T g). \tag{81}$$

We will bound the expected objective value $\mathbb{E}[x^T C x]$ of the rounded solution. Here, the expected value is taken over a standard Gaussian random vector $g \in \mathbb{R}^n$. This will be aided by two short calculations, valid for any two unit vectors $c, b \in \mathbb{R}^n$. First,

$$\mathbb{E}[b^T g g^T c] = b^T \mathbb{E}[g g^T] c = b^T c.$$

Next, using the fact that the distribution of the Gaussian vector is rotationally invariant, we may assume that $c = e_1$ is equal to the first element of the standard basis, and $b = b_1\, e_1 + b_2\, e_2$ is a linear combination of the first two basis vectors. Similarly, g_1 and g_2 denote the first two components of g. Then

$$\begin{aligned}
\mathbb{E}[(b^T g)\operatorname{sign}(c^T g)] &= \mathbb{E}[(b_1 g_1 + b_2 g_2)\operatorname{sign}(g_1)] \\
&= \mathbb{E}[b_1 g_1 \operatorname{sign}(g_1)] + \mathbb{E}[b_2 g_2] \underbrace{\mathbb{E}[\operatorname{sign}(g_1)]}_{=0} \\
&= b_1 \mathbb{E}[g_1 \operatorname{sign}(g_1)] \\
&= b_1 \frac{2}{\sqrt{2\pi}} \int_0^\infty x e^{-x^2/2}\, \mathrm{d}x = \sqrt{\frac{2}{\pi}}\, b^T c.
\end{aligned}$$

From this, we can expand

$$\begin{aligned}
\frac{\pi}{2}\mathbb{E}\left[x_i x_j\right] =& \frac{\pi}{2}\mathbb{E}\left[\operatorname{sign}(u_i^T g)\operatorname{sign}(u_j^T g)\right] \\
=& \underbrace{\mathbb{E}\left[\left(u_i^T g - \sqrt{\frac{\pi}{2}}\operatorname{sign}(u_i^T g)\right)\left(u_j^T g - \sqrt{\frac{\pi}{2}}\operatorname{sign}(u_j^T g)\right)\right]}_{=:\Delta_{ij}} \\
& - \mathbb{E}[u_i^T g\, g^T u_j] + \sqrt{\frac{\pi}{2}}\mathbb{E}[u_i^T g\, \operatorname{sign}(u_j^T g)] + \sqrt{\frac{\pi}{2}}\mathbb{E}[u_j^T g\, \operatorname{sign}(u_i^T g)] \\
=& \Delta_{ij} - u_i^T u_j + u_i^T u_j + u_i^T u_j = \Delta_{ij} + n\sigma_{ij}.
\end{aligned}$$

As a convex combination of symmetric rank-one matrices, Δ is psd. Factoring out, using the two calculations above, and as well as $u_i^T u_i = n\sigma_{ii} = 1$, we see that its main diagonal elements satisfy

$$\begin{aligned}\Delta_{ii} &= \mathbb{E}\left[\left(u_i^T g - \sqrt{\frac{\pi}{2}}\operatorname{sign}(u_i^T g)\right)\left(u_i^T g - \sqrt{\frac{\pi}{2}}\operatorname{sign}(u_i^T g)\right)\right] \\ &= (1-2)u_i^T u_i + \frac{\pi}{2} = \frac{\pi}{2} - 1.\end{aligned} \tag{82}$$

Thus, $(n(\frac{\pi}{2}-1))^{-1}\Delta$ is feasible for the SDP. For a cost matrix with the given block form, the spectrum of objective values for the SDP is symmetric. This can be seen by considering the block matrix

$$U = \begin{pmatrix} \mathbb{1} & 0 \\ 0 & -\mathbb{1} \end{pmatrix}. \tag{83}$$

Then, $U\rho^\star U^\dagger$ is also feasible for the SDP and

$$\operatorname{tr}(CU\rho^\star U^\dagger) = \operatorname{tr}(U^\dagger CU\rho^\star) = -\operatorname{tr}(C\rho^\star). \tag{84}$$

Thus, we know that $-\operatorname{tr}(C\rho^\star)$ is a lower bound for the SDP and therefore,

$$|\operatorname{tr}(C\Delta)| \leq \left(\frac{\pi}{2} - 1\right) n \operatorname{tr}(C\rho^\star). \tag{85}$$

This allows us to bound

$$\mathbb{E}\left[x^T C x\right] = \frac{2}{\pi}\left(\operatorname{tr}(Cn\sigma) + \operatorname{tr}(C\Delta)\right) \tag{86}$$

$$\geq \frac{2}{\pi}\left(\operatorname{tr}(Cn\sigma) - |\operatorname{tr}(C\Delta)|\right) \tag{87}$$

$$\geq \frac{2}{\pi} n \operatorname{tr}(C\sigma) - \left(1 - \frac{2}{\pi}\right) n \operatorname{tr}(C\rho^\star). \tag{88}$$

□

The proofs of Lem. 12 and Thm. 5 are based on Ref. [25, Sec. 3.5]. They are adjusted to also consider the improved scaling from Thm. 4.

Lemma 12. *Let $0 \leq \xi \leq 1/2$ and $\rho \in \mathbb{R}^{n\times n}$ be a psd matrix such that $\sum_i |\rho_{ii} - 1/n| \leq \xi^3$. Let $\sigma \in \mathbb{R}^{n\times n}$ be the matrix with entries $\sigma_{ij} = \frac{\rho_{ij}}{n\sqrt{\rho_{ii}\rho_{jj}}}$. Then,*

$$\|\rho - \sigma\|_{\mathrm{tr}} = \mathcal{O}(\xi). \tag{89}$$

Proof. Similar to the proof of Thm. 4, we define the index set $B = \{i : |d_i| > \frac{\xi}{n}\}$ with $d_i = \rho_{ii} - 1/n$ and write σ as block matrix

$$\sigma = \begin{pmatrix} \sigma^{(11)} & \sigma^{(12)} \\ \sigma^{(21)} & \sigma^{(22)} \end{pmatrix}, \tag{90}$$

where $\sigma^{(22)}$ corresponds to the index set of $\bar{B}$.

We now show that $\left\|\rho^{(22)} - \sigma^{(22)}\right\|_{\mathrm{tr}} = O(\xi)$. To this end, define the diagonal $|\bar{B}| \times |\bar{B}|$-matrix by

$$(D_{\bar{B}})_{ii} = \frac{1}{\sqrt{n\rho_{ii}}}, \qquad i \in \bar{B}. \tag{91}$$

With this we can define a linear map $\mathcal{D} : X \mapsto D_{\bar{B}} X D_{\bar{B}}$ on the space of $|\bar{B}| \times |\bar{B}|$-matrices that fulfills $\mathcal{D}(\rho^{(22)}) = \sigma^{(22)}$.

We can then bound

$$\left\|\rho^{(22)} - \sigma^{(22)}\right\|_{\mathrm{tr}} = \left\|(\mathbb{1} - \mathcal{D})\rho^{(22)}\right\|_{\mathrm{tr}} \tag{92}$$

$$\leq \|\mathbb{1} - \mathcal{D}\|_{\mathrm{tr}\to\mathrm{tr}} \left\|\rho^{(22)}\right\|_{\mathrm{tr}} \tag{93}$$

$$\leq \|\mathbb{1} - \mathcal{D}\|_{\mathrm{tr}\to\mathrm{tr}}\,, \tag{94}$$

because $\rho^{(22)}$ is a submatrix of ρ and $\|\rho\|_{\mathrm{tr}} = 1$. As both $\mathbb{1}$ and $\mathcal{D}$ are self-adjoint with respect of the Frobenius inner product $\mathrm{tr}(X^T Y)$, the dualtity of norms implies $\|\mathbb{1} - \mathcal{D}\|_{\mathrm{tr}\to\mathrm{tr}} = \|\mathbb{1} - \mathcal{D}\|_{\infty\to\infty}$.

By the definition of B, the diagonal elements of $\rho^{(22)}$ lie in $[(1-\xi)/n, (1+\xi)/n]$. Then

$$\frac{1}{\sqrt{1+\xi}} \leq (D_{\bar{B}})_{ii} \leq \frac{1}{\sqrt{1-\xi}}, \tag{95}$$

and thus for $\xi \leq 1/2$:

$$1 - \xi \leq (D_{\bar{B}})_{ii} \leq 1 + \xi. \tag{96}$$

Next, set $D_\xi = D_{\bar{B}} - \mathbb{1}$. From Eq. (96) it then follows that $\|D_\xi\| \leq \xi$. Now,

$$\|(\mathbb{1} - \mathcal{D})(X)\|_\infty = \|X D_\xi + D_\xi X + D_\xi X D_\xi\|_\infty \tag{97}$$

$$= 2\,\|D_\xi\|_\infty \|X\|_\infty + \|D_\xi\|_\infty^2 \|X\|_\infty \tag{98}$$

$$\leq 3\xi\,\|X\|_\infty\,, \tag{99}$$

and hence $\|\mathbb{1} - \mathcal{D}\|_{\infty\to\infty} \leq 3\xi$. We therefore have

$$\left\|\rho^{(22)} - \sigma^{(22)}\right\|_{\mathrm{tr}} \leq \|\mathbb{1} - \mathcal{D}\|_{\mathrm{tr}\to\mathrm{tr}} = \|\mathbb{1} - \mathcal{D}\|_{\infty\to\infty} = \mathcal{O}(\xi). \tag{100}$$

Next, we bound the remaining blocks of σ using a similar analysis as in Eqs.(71)-(75) in the proof of Thm. 4. We use that σ is psd and its diagonal entries are equal to $1/n$. Then,

$$\left\|\sigma^{(22)}\right\|_{\mathrm{tr}} = \mathrm{tr}(\sigma^{(22)}) = 1 - \frac{|B|}{n} = 1 - \mathcal{O}(\xi^2), \tag{101}$$

and thus $\sigma^{(11)} + 2\sigma^{(12)} = \mathcal{O}(\xi)$. Additionally, from Eq. (75) we have $\rho^{(11)} + 2\rho^{(12)} = \mathcal{O}(\xi)$. Combining the above, we have

$$\begin{aligned}\|\rho - \sigma\|_{\mathrm{tr}} \leq& \left\|\sigma^{(22)} - \rho^{(22)}\right\|_{\mathrm{tr}} + \left\|\rho^{(11)}\right\|_{\mathrm{tr}} + 2\left\|\rho^{(12)}\right\|_{\mathrm{tr}} + \left\|\sigma^{(11)}\right\|_{\mathrm{tr}} + 2\left\|\sigma^{(12)}\right\|_{\mathrm{tr}} \\ =& \mathcal{O}(\xi)\end{aligned}$$

□

Theorem 5. *Let $\rho^\star \in \mathbb{R}^{n\times n}$ be the optimal SDP solution corresponding to a normalized cost matrix $C \in \mathbb{R}^{n\times n}$ with a block structure as defined in* (3)*. Let $\rho \in \mathbb{R}^{n\times n}$ be an approximate solution with*

$$\operatorname{tr}(C\rho^\star) - \operatorname{tr}(C\rho) \leq \epsilon, \tag{39}$$

$$\text{and} \qquad \sum_i |\rho_{ii} - \frac{1}{n}| \leq \epsilon \qquad \forall i \in [n], \tag{40}$$

with $0 \leq \epsilon \leq 1/2$. Let $x \in \{-1, 1\}^n$ be the vector obtained by applying the randomized rounding procedure to ρ. Then

$$\mathbb{E}[x^T C x] \geq \left(\frac{4}{\pi} - 1\right) n \operatorname{tr}(C\rho^\star) - \mathcal{O}(n\epsilon^{1/3}). \tag{41}$$

Proof. Set $\xi = \epsilon^{1/3}$ and $\sigma_{ij} = \frac{\rho_{ij}}{n\sqrt{\rho_{ii}\rho_{jj}}}$ for $i, j = 1, \ldots, n$. From the assumptions we have $\|C\| = 1$. Then, using Hölder's inequality and applying Lem. 12 gives

$$|\operatorname{tr}(C(\rho - \sigma))| \leq \|\rho - \sigma\|_{\mathrm{tr}} = \mathcal{O}(\xi) \tag{102}$$

and therefore $|\operatorname{tr}(C(\rho^\star - \sigma))| \leq \mathcal{O}(\xi) + \epsilon = \mathcal{O}(\epsilon^{1/3})$. Now, applying Lem. 11 gives

$$\mathbb{E}\left[x^T C x\right] \geq \frac{2}{\pi} n \operatorname{tr}(C\sigma) - \left(1 - \frac{2}{\pi}\right) n \operatorname{tr}(C\rho^\star) \tag{103}$$

$$\geq \frac{2}{\pi} n \left(\operatorname{tr}(C\rho^\star) - \mathcal{O}(\epsilon^{1/3})\right) - \left(1 - \frac{2}{\pi}\right) n \operatorname{tr}(C\rho^\star) \tag{104}$$

$$= \left(\frac{4}{\pi} - 1\right) n \operatorname{tr}(C\rho^\star) - \mathcal{O}(\epsilon^{1/3} n). \tag{105}$$

□

B Number of quantum gates for Hamiltonian Updates

In Sec. 5, we gave a procedure for approximating the diagonal entries of a Gibbs state for given Hamiltonian, and its asymptotic scaling. Now, we will give lower bounds on the number of two-qubit gates needed for an iteration of HU with selected quantum algorithms [5, 34, 33]. These consists of block encoding, Hamiltonian simulation and Gibbs state preparation. We assume a gate model, where two-qubit gates are realized by CNOT gates.

B.1 Block encoding

This section is based on Ref. [33], and reproduces some of the their construction in order to make the present document self-contained.

The input is a Hamiltonian $H \in \mathbb{R}^{n\times n}$. We assume that it is s-sparse in the sense that exactly s elements in each column of H are explicitly specified, with the rest being equal to 0. (We refer to the specified elements as the *non-zero* ones, though "potentially non-zero" would be more accurate).

The matrix is assumed to be stored in QRAM, accessible via two unitary quantum oracles:

- O_F takes two indices $i \in [n]$, $l \in [s]$ and maps them to the index of the l^{th} non-zero entry in the i^{th} row of H, denoted by $f(i,l)$:

$$O_F|i\rangle|l\rangle = |i\rangle|f(i,l)\rangle. \tag{106}$$

- O_H takes two indices $i, j \in [n]$ and returns the matrix entry H_{ij} represented as a b-bit number:

$$O_H|i\rangle|j\rangle|z\rangle = |i\rangle|j\rangle|z \oplus H_{ij}\rangle. \tag{107}$$

Definition 13 ([33, Def. 6])**.** *A unitary U* block-encodes *a Hamiltonian H if*

$$U = \begin{pmatrix} H/\alpha & \cdot \\ \cdot & \cdot \end{pmatrix}, \quad (\langle 0|_a \otimes \mathbb{1}_d) U (|0\rangle_a \otimes \mathbb{1}_d) = \frac{H}{\alpha}, \tag{108}$$

where d is a b-qubit register and α a suitable scaling factor.

In the following encoding α is bounded from below by $s\,\|H\|_{\max}$.
Reference [33, Thm. 10] realize a block encoding U with the following properties:

- [33, Lem. 13] The encoding U is decomposed as a product $U = U_{\text{row}}^\dagger U_{\text{col}}$.
- [33, Lem. 13] Both U_{row} and U_{col} consist of one invocation each of O_F, O_H, O_F^{-1}, as well as a further unitary. Additionally, U_{col} uses two controlled SWAPs on $\log_2(n)$ qubits each.
- [33, Lem. 15] This additional unitary is a product of single-qubit gates and an b-bit comparer.

To lower-bound the number of two-qubit gates used in this construction, we use the results of Ref. [20]. They show that a b-bit comparer can be implemented using $(2b-1)$ Toffoli gates and $4b-3$ CNOTs. In turn, each Toffoli may be realized using 6 CNOTs and a number of single-qubit gates [44]. For further steps, we need a controlled implementation of U. This can be achieved by just controlling the SWAP gates, as without these, the quantum circuit just simplifies to the identity (c.f. Fig. 7). A controlled SWAP on $\log_2(n)$ qubits can be realized with $\log_2(n)$ Tofolli gates and $2\log_2(n)$ CNOTs.

Estimate 14. *The controlled implementation of the block encoding U given in Ref. [33] requires at least $16b + 16\log_2(n) - 9$ two-qubit gates.*

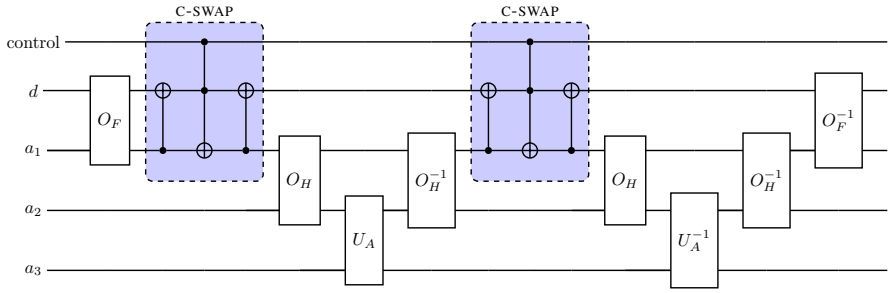

Figure 7: Circuit for a controlled U, resulting by combining U_{col} and $U_{\mathrm{row}}^{\dagger}$. The SWAP gates are part of U_{col}, controlling them allows to control the whole circuit of U. The registers d and a_1 encode the index of row and column condisting of $\log_2(n)$ qubits each, a_2 encodes the corresponding entry of H with b qubits, and a_3 is an $O(1)$ qubit ancillary register.

B.2 Hamiltonian simulation via Qubitization

The Hamiltonian simulation uses a procedure called *qubitization*, which is explained in Ref. [34]. It has the following properties:

- [34, Thm .1, Lem. 6] The procedure simulates an encoded Hamiltonian $\frac{H}{s\|H\|_{\max}}$ for some time t.
- [34, Sec. 5.2, Thm. 4] The circuit consists of Q unitaries V_i, where Q is the smallest integer $Q = q - 1$ s.t.

$$\epsilon \geq \frac{4t^q}{2^q q!}. \tag{109}$$

- [34, Lem. 10] Each V_i consists of both the controlled unitaries U and U^{-1} defined in Sec. B.1, besides further single- and two-qubit gates.

Equation (109) can be written as

$$q + \log_2(q!) + \log_2(\epsilon) \geq 2 + q\log_2(t). \tag{110}$$

To simplify this expression, we use an upper bound on Stirling's approximation

$$\log_2(q!) \leq q\log_2(q) - q\log_2(e) + \frac{1}{2}\log_2(q) + \log_2(\sqrt{2\pi}) + \frac{1}{12}\log_2(e). \tag{111}$$

where e is Euler's number. Inserting (111) into (110) gives

$$q\log_2(q) + q(1 - \log_2(e)) + \log_2(\epsilon) + 1.5 \geq q + \log_2(q!) + \log_2(\epsilon) \geq 2 + q\log_2(t), \tag{112}$$

and thus,

$$q(\log_2(q) - \log_2(t) + 1 - \log_2(e)) \geq 0.5 - \log_2(\epsilon). \tag{113}$$

As the right-hand side is positive, we require at least,

$$\log_2(q) - \log_2(t) + 1 - \log_2(e) \geq 0, \tag{114}$$

and thus, we can give a lower bound on a q fulfilling Eq. (109):

$$q \geq 0.73t. \tag{115}$$

As mentioned above, this circuit only simulates the normalized Hamiltonian $\frac{H}{s\|H\|_{\max}}$ for time t. We simulate H for some time τ by setting $t = s\,\|H\|_{\max}\,\tau$.

Estimate 15. *Simulating H for some time τ using the above construction requires at least Q implementations of U and U^{-1} each. The required number of two-qubit gates is at least*

$$\begin{aligned} C_H(\tau,\epsilon) &= (32b + 32\log_2(n) - 18)\,Q \\ &\geq (32b + 32\log_2(n) - 18)\,(0.73s\,\|H\|_{\max}\,\tau - 1). \end{aligned} \tag{116}$$

B.3 Gibbs state sampling

We mainly focus on the estimation of the diagonal of ρ as this is $O(n)$ times more expensive than computing a trace product $\mathrm{tr}(A\rho)$. For this part of the gate count we only consider the gates that result from the Hamiltonian simulation subroutine described above and neglect the additional overhead. The following is based on Ref. [5]:

As described in Sec. 5, we need to compute an estimate $\tilde{\lambda}_{\min}$ of the smallest eigenvalue of H up to additive error $1/2$. This needs to be done only once per H and thus does not influence the complexity. Using this, we implement a modified version of O_H, denoted as O_{H_+} with $H_+ = H - (\tilde{\lambda}_{\min} - 3/2)\mathbb{1}$, and denote by $C_{H_+}(\tau,\epsilon)$ the corresponding lower bound on the number of two-qubit gates for simulating H_+ with the given construction analogous to Eq. (116).

Definition 16 ([5, Def. 35])**.** *We call $\tilde{W}$ a controlled (M,γ,ϵ)-simulation of H if*

$$\left\|\tilde{W} - W\right\| \leq \epsilon, \tag{117}$$

where

$$W := \sum_{m=-M}^{M-1} |m\rangle\langle m| \otimes e^{im\gamma H} \tag{118}$$

is a controlled simulation of H.

- [5, Lem. 36] The implementation of a controlled (M, γ, ϵ)-simulation of H_+ using the construction of Ref. [5] requires at least

$$C_{\mathrm{cont}-(M,\gamma,\epsilon)} = \sum_{j=0}^{\log_2(M)} C_{H_+}(2^j\gamma, 2^{j-\log_2(M)-1}\epsilon) \geq C_{H_+}(M\gamma, \epsilon/2) \quad (119)$$

  two-qubit gates.

Lemma 17 ([5, Thm. 43])**.** *Computing the sub normalized Gibbs state e^{-H_+} to ϵ-precision using the constructions in Ref. [5, 34, 33] requires at least*

$$C_{(e^{-H_+})} \geq C_{H_+}(2\pi \ln(7.8\epsilon^{-1}), \epsilon/4) \quad (120)$$

two-qubit gates.

Proof. The constants for the non-asymptotic scaling in Lem. 17 are retrieved from the proofs of Ref. [5, Lem. 37, Thm. 40, 43]:

- Ref. [5, Thm. 43] requires $\|H_+\| \leq 2r$, $\delta = 1/2$.
- Ref. [5, Thm. 43] defines $f(x + x_0) = \sum_{l=0}^{\infty} a_l x^l = e^{x-x_0}$, $x_0 = r + \delta$. Thus, $a_l = e^{-x_0}\frac{(-1)^l}{l!}$.
- Ref. [5, Thm. 40] defines $g\left(\frac{x}{r+\delta}\right) = f(x + x_0)$ and $g(y) = \sum_{l=0}^{\infty} b_l y^l$. Thus, $b_l = a_l(r + \delta)^l$.
- Ref. [5, Thm. 40] sets $L = \lceil \frac{1}{\delta'} \ln(8/\epsilon) \rceil$ with $\delta' = \delta/(r + \delta)$ and defines the L-truncated polynomial approximation of g and it's corresponding M-truncated Fourier approximation given by [5, Lem. 37].
- Ref. [5, Lem. 37] requires

$$M = \max\left(2\lceil \ln\left(\frac{4\|b\|_1}{\epsilon'}\right)\frac{1}{\delta'}\rceil, 0\right), \quad (121)$$

  where

$$\|b\|_1 = \sum_{l=0}^{L} |b_l| = e^{-x_0}\sum_{l=0}^{L}\frac{x_0^l}{l!}. \quad (122)$$

- Ref. [5, Thm. 40] requires a controlled $(M, \gamma = \frac{\pi}{2r+2\delta}, \frac{\epsilon}{2})$ simulation.

We can directly form an upper bound for δ':

$$\delta' = \frac{\frac{1}{2}}{r + \frac{1}{2}} \leq \frac{1}{\|H_+\| + 1}. \quad (123)$$

We now want to find a lower bound for $\|b\|_1$. We start with rewriting Eq. (122):

$$\begin{aligned}\|b\|_1 &= e^{-x_0}\left(\sum_{l=0}^{\infty}\frac{x_0^l}{l!}-\sum_{l=L+1}^{\infty}\frac{x_0^l}{l!}\right)\\ &= 1-e^{-x_0}\sum_{l=L+1}^{\infty}\frac{x_0^l}{l!}\\ &= 1-e^{-x_0}\sum_{t=0}^{\infty}\frac{x_0^{L+1+t}}{(L+1+t)!}.\end{aligned} \tag{124}$$

Next, we use

$$L+1 \geq \lceil \frac{1}{\delta'}\ln(8/\epsilon)\rceil \geq (2r+1)\ln(8/\epsilon) = 2x_0\ln(8/\epsilon), \tag{125}$$

and therefore, $x_0 < (L+2)/2$. This allows us to compare Eq. (124) to a geometric series:

$$\begin{aligned}\|b\|_1 &\geq 1-e^{-x_0}\frac{x_0^{L+1}}{(L+1)!}\sum_{t=0}^{\infty}\left(\frac{x_0}{L+2}\right)^t\\ &\geq 1-e^{-x_0}\frac{x_0^{L+1}}{(L+1)!}\sum_{t=0}^{\infty}\left(\frac{1}{2}\right)^t\\ &= 1-2e^{-x_0}\frac{x_0^{L+1}}{(L+1)!}.\end{aligned} \tag{126}$$

Next, we use a lower bound on Stirling's approximation

$$z! \geq e^{z\ln(z)-z}, \tag{127}$$

giving us

$$\begin{aligned}\|b\|_1 &\geq 1-2e^{-x_0+\ln(x_0)(L+1)+(L+1)-\ln(L+1)(L+1)},\\ &= 1-2e^{-x_0-(L+1)\left(\ln\left(\frac{L+1}{x_0}\right)-1\right)}.\end{aligned} \tag{128}$$

Then, with $x_0 \geq 1/2$ and $\epsilon \leq 1$,

$$\ln\left(\frac{L+1}{x_0}\right)-1 \geq \ln(2\ln(8))-1, \tag{129}$$

and thus,

$$\|b\|_1 \geq 1-2e^{-1/2-2\ln(8)(2\ln(8)-1)} \geq 0.98. \tag{130}$$

Now, that we know a lower bound on $\|b\|_1$, we can finally bound M using (121) and (123):

$$M \geq 2\ln\left(\frac{4\|b\|_1}{\epsilon'}\right)\frac{1}{\delta'} \geq 2(\|H_+\|+1)\ln(7.8\epsilon^{-1}). \tag{131}$$

With $\gamma = \frac{\pi}{2r+2\delta} = \frac{\pi}{\|H_+\|+1}$, we thus have

$$M\gamma \geq 2\pi \ln(7.8\epsilon^{-1}). \tag{132}$$

Then combining Eq. (119) and Lem. 17 gives

$$C_{(e^{-H_+})} \geq C_{\mathrm{cont}-(M,\gamma,\epsilon/2)} \geq C_{H_+}(2\pi \ln(7.8\epsilon^{-1}), \epsilon/4). \tag{133}$$

□

The procedure above gives a sub normalized Gibbs state e^{-H_+}. To obtain a normalized one *amplitude amplification* is applied to increase the amplitude by a factor $1/\operatorname{tr}(e^{-H_+})$. The expected number of amplitude amplification iterations using the construction in Ref. [14, Sec. 3] is $0.69\sqrt{n/z}$, where z is a lower bound of $\operatorname{tr}(e^{-H_+})$. By our preparation of H_+ we ensured that its minimum eigenvalue is $\lambda_{\min} \leq 2$ and thus $z = e^{-2}$ suffices. Thus, the expected cost for preparing the Gibbs state is

$$C_\rho = 0.69e\, n^{1/2} C_{(e^{-H_+})} \geq 1.87 n^{1/2} C_{H_+}(2\pi \ln(7.8\epsilon^{-1}), \epsilon/4). \tag{134}$$

Finally, combining Eq. (134) and Obs. 15 gives the following conservative estimate:

Estimate 7. *Let $H \in \mathbb{R}^{n\times n}$ be an s-sparse matrix and b be the number of bits per entry used to store H. The expected number of two-qubit gates used to prepare an approximation of the Gibbs state ρ_H with precision ϵ using the constructions in Refs. [33, 20, 34, 5, 14] is at least*

$$(32b + 32\log_2(n) - 18)\,(4.5\ln(7.8\epsilon^{-1})\, n^{1/2} s\, \|H_+\|_{\max} - 1), \tag{44}$$

with H_+ constructed from H as defined in Eq. (43)*.*

B.4 Diagonal entry sampling

We now derive the result of Estimate 8 that states the required number of Gibbs state preparations on a quantum computer needed to estimate the diagonal entries. The diagonal entries are estimated statistically by measuring the Gibbs states in the computational basis. For technical reasons, we randomize the total number of samples according to a Poisson distribution $\mathrm{Pois}(m)$, where m is the expected number of samples needed. Then, our estimate for each diagonal entry is given by $\hat{\rho}_{ii} = N_i/m$, where N_i represents the number of measurements resulting in the i^{th} state. Note that we are dividing by the expected number of measurements, not the actual one.

Lemma 18. *Let $\eta \leq 1/8$. Given N preparations of $\tilde{\rho} \in \mathbb{R}^{n\times n}$, where N is drawn from a Poisson distribution with mean $m = 2.13\eta^{-2}(n\ln 2 + \ln(1/p))$, we can get estimates $\{\hat{\rho}_{ii}\}_{i\in\{1,\ldots,n\}}$ s.t. $\sum_i |\tilde{\rho}_{ii} - \hat{\rho}_{ii}| \leq \eta$ with probability $1-p$.*

Proof. Let N_i be the number of times the outcome i was obtained, so that the total N satisfies $N = \sum_i N_i$. The randomization of N makes the N_i independent:

$$\begin{aligned}\Pr[N_i = x_i \,\forall i \in [n]] &= \frac{m^{\sum_i x_i} e^{-m}}{(\sum_i x_i)!}(\sum_i x_i)! \prod_i \frac{p_i^{x_i}}{x_i!} \\ &= \prod_i \frac{(mp_i)^{x_i} e^{-mp_i}}{x_i!} \\ &= \prod_i \Pr[\mathrm{Pois}(mp_i) = x_i].\end{aligned} \tag{135}$$

The moment generating function for N_i is given by

$$M_{N_i}(\lambda) = \mathbb{E}\left[e^{\lambda N_i}\right] = \exp(mp_i(e^\lambda - 1)). \tag{136}$$

Define $\delta_i = \hat{p}_i - p_i$. Then

$$\begin{aligned}M_{\delta_i}(\lambda) = \mathbb{E}\left[e^{\lambda\delta_i}\right] &= \mathbb{E}\left[e^{\lambda(N_i/m - p_i)}\right] \\ &= e^{-\lambda p_i}\mathbb{E}\left[e^{\lambda/mN_i}\right] \\ &= e^{-\lambda p_i} M_{N_i}(\lambda/m) \\ &= \exp(-\lambda p_i + mp_i(e^{\lambda/m} - 1)).\end{aligned} \tag{137}$$

This moment generating function fulfills $M_{\delta_i}(|\lambda|) \geq M_{\delta_i}(\lambda)$. Indeed, for $\lambda > 0$ a series expansion gives

$$\begin{aligned}\ln M_{\delta_i}(\lambda) &= -\lambda p_i + mp_i(e^{\lambda/m} - 1) \\ &= mp_i \sum_{k=2}^{\infty} \frac{(\lambda/m)^k}{k!} \\ &\geq mp_i \sum_{k=2}^{\infty} \frac{(-\lambda/m)^k}{k!} = \ln M_{\delta_i}(-\lambda).\end{aligned} \tag{138}$$

Then, arguing as in the standard proof of the Chernoff bound,

$$\begin{aligned}
\Pr\left[\sum_{i=1}^{n}|\delta_i| \geq \eta\right] &\leq e^{-\lambda\eta}\mathbb{E}\left[e^{\lambda\sum_{i=1}^{n}|\delta_i|}\right] \\
&= e^{-\lambda\eta}\prod_{i=1}^{n}\mathbb{E}\left[e^{\lambda|\delta_i|}\right] \\
&\leq e^{-\lambda\eta}\prod_{i=1}^{n}\mathbb{E}\left[e^{\lambda\delta_i}+e^{-\lambda\delta_i}\right] \\
&\leq e^{-\lambda\eta}\prod_{i=1}^{n}2M_{\delta_i}(\lambda) \\
&= 2^n\exp\left(-\lambda\eta-\lambda\sum_{i=1}^{n}p_i+m\sum_{i=1}^{n}p_i(e^{\frac{\lambda}{m}}-1)\right) \\
&= 2^n\exp\left(-\lambda(\eta+1)+m(e^{\frac{\lambda}{m}}-1)\right).
\end{aligned} \tag{139}$$

Choosing $\lambda = m\ln(1+\eta)$ gives us

$$\Pr\left[\sum_{i=1}^{n}|\delta_i| \geq \eta\right] \leq 2^n e^{m(\eta-(1+\eta)\ln(1+\eta))}. \tag{140}$$

Then, with $\ln(1+\eta) \geq \eta - \frac{\eta^2}{2} + \frac{29}{96}\eta^3$ for $\eta \leq \frac{1}{8}$,

$$\begin{aligned}
\Pr\left[\sum_{i=1}^{n}|\delta_i| \geq \eta\right] &\leq 2^n e^{m(\eta-(1+\eta)(\eta-\frac{\eta^2}{2}+\frac{29}{96}\eta^3))} \\
&= 2^n e^{m(-\frac{\eta^2}{2}+(\frac{1}{2}-\frac{29}{96})\eta^3+\frac{29}{96}\eta^4)} \\
&\leq 2^n e^{-\frac{m\eta^2}{2.13}}.
\end{aligned} \tag{141}$$

Demanding $\Pr\left[\sum_{i=1}^{n}|\delta_i| \geq \eta\right] \leq p$ and solving for m concludes the proof. □

Lemma 19. *Let $\epsilon \leq 1/4$. Given N approximate preparations $\tilde{\rho}$ of ρ on a quantum computer with $\sum_i|\tilde{\rho}_{ii}-\rho_{ii}| \leq \frac{\epsilon}{8}$, where N is drawn from a Poisson distribution with mean $m = 137\epsilon^{-2}(n\ln(2)+\ln(1/p))$, we can obtain classical estimates $\hat{\rho}_{ii}$ s.t. $\sum_i|\tilde{\rho}_{ii}-\hat{\rho}_{ii}| \leq \frac{\epsilon}{4}$ with probability $1-p$.*

Proof. Using Lem. 18 with $\eta = \epsilon/8$ gives $\sum_i|\hat{\rho}_{ii}-\tilde{\rho}_{ii}| \leq \frac{\epsilon}{8}$. Then, by triangular inequality we get

$$\sum_i|\hat{\rho}_{ii}-\rho_{ii}| \leq \sum_i|\hat{\rho}_{ii}-\tilde{\rho}_{ii}| + \sum_i|\tilde{\rho}_{ii}-\rho_{ii}| \leq \frac{\epsilon}{8}+\frac{\epsilon}{8} = \frac{\epsilon}{4}. \tag{142}$$

□

For small ϵ the required mean number of samples approaches

$$m = 128\epsilon^{-2}(n\ln(2)+\ln(1/p)). \tag{143}$$

For the benchmarking, we assume $\ln(1/p)) \geq 4$.

B.5 Quantum HU algorithm

Here we give the high-level HU routine that uses quantum subroutines for estimates involving the Gibbs state ρ. The quantum algorithms are discussed in Sec. 5.

Algorithm 4 Hamiltonian Updates with a quantum computer

Require: Query access to $\epsilon/4$-precise subroutines

$$\text{QC\_GIBBS\_TRACE\_PRODUCT} : H, A \in \mathbb{R}^{n\times n} \mapsto \operatorname{tr}(A\rho_H),$$
$$\text{QC\_GIBBS\_DIAGONALS} : H \in \mathbb{R}^{n\times n} \mapsto \{(\rho_H)_{ii}\}_{i=1,\dots,n},$$

normalized cost matrix $C \in \mathbb{R}^{n\times n}$, threshold objective value γ, precision parameter ϵ, initial step lengths λ_c and λ_d, momentum hyperparameter β

Ensure:	**Condition**	**Output**
	(5) is feasible	H, such that ρ_H is ϵ-feasible
	(5) is not ϵ-feasible	false
	else	undefined (H, such that ρ_H is ϵ-feasible, or false)

function QUANTUM_HAMILTONIAN_UPDATES($C, \gamma, \epsilon, \lambda_c, \lambda_d, \beta$)
  $P_c \leftarrow -C + \gamma\mathbb{1}$
  $H \leftarrow 0_{n\times n}$
  $M \leftarrow 0_{n\times n}$
  $F = -\ln(n)$

  while $F \leq 0$ **do**
    $\operatorname{tr}(P_c\rho) \leftarrow$ QC_GIBBS_TRACE_PRODUCT(H, P_c)
    if $\operatorname{tr}(P_c\rho) > \frac{3}{4}\epsilon$ **then**
      $\Delta H \leftarrow \operatorname{tr}(P_c\rho)P_c + \frac{\beta}{\lambda_c}M$
      $H, F, \lambda_c \leftarrow$ QUANTUM_UPDATE$(H, \Delta H, F, \epsilon, \lambda_c)$ ▷ Cost update
      $M \leftarrow \lambda_c\Delta H$
      continue
    end if

    $\{\rho_{ii}\} \leftarrow$ QC_GIBBS_DIAGONALS(H) ▷ Estimate diagonal of ρ on QC
    if $\sum_i |\rho_{ii} - 1/n| > \frac{3}{4}\epsilon$ **then**
      $P_d^{\ell_2} \leftarrow \sum_i(\rho_{ii} - 1/n)|i\rangle\langle i|/\max_i|\rho_{ii} - 1/n|$
      $\Delta H \leftarrow P_d^{\ell_2} + \frac{\beta}{\lambda_d}M$
      $H, F, \lambda_d \leftarrow$ QUANTUM_UPDATE$(H, \Delta H, F, \epsilon, \lambda_d)$ ▷ Diag. update
      $M \leftarrow \lambda_d\Delta H$
      continue
    end if

    return H
  end while
  return false
end function

Algorithm 5 Update function with a quantum computer

Require: Query access to $\epsilon/4$-precise subroutine

$$\text{QC\_GIBBS\_TRACE\_PRODUCT} : H, A \in \mathbb{R}^{n\times n} \mapsto \text{tr}(A\rho_H),$$

Hamiltonian H, update matrix ΔH, free energy bound F, precision parameter ϵ, current step length λ_c or λ_d, number of trace estimations for free energy bound J

Ensure: Updated Hamiltonian H_{new}, updated free energy bound F, updated step length λ_c or λ_d

function QUANTUM_UPDATE($H, \Delta H, F, \epsilon, \lambda$)
    $H_{\text{new}} \leftarrow H + \lambda\Delta H$

    $\text{tr}(\Delta H\rho_{\text{new}}) \leftarrow$ QC_GIBBS_TRACE_PRODUCT$(H_{\text{new}}, \Delta H)$
    while $\text{tr}(\Delta H\rho_{\text{new}}) < \epsilon/4$ **do**: ▷ Check for overshoots; finite precision
        $\lambda \leftarrow 0.5\lambda$
        $H_{\text{new}} \leftarrow H + \lambda\Delta H$
        $\text{tr}(\Delta H\rho_{\text{new}}) \leftarrow$ QC_GIBBS_TRACE_PRODUCT$(H_{\text{new}}, \Delta H)$
    end while

    for $j \in [J]$ **do**: ▷ Estimate the change in free energy
        $\text{tr}(\rho_{H+\frac{j}{J}\lambda\Delta H}\Delta H) \leftarrow$ **qc_gibbs_trace_product**$(H + \frac{j}{J}\lambda\Delta H, \Delta H)$
        $F \leftarrow F + \frac{1}{J}\left(\text{tr}(\rho_{H+\frac{j}{J}\lambda\Delta H}\Delta H) - \frac{\epsilon}{4}\right)$ ▷ Account for finite precision
    end for

    $\lambda \leftarrow 1.3\lambda$

    return $H_{\text{new}}, F, \lambda$
end function

Chapter 5

Benchmarking of SDP relaxations for real-world problems

In this chapter, we analyze the performance of the HU algorithm in combination with the GW method for solving real-world optimization problems. In particular, we investigate how the quality of the obtained solutions depends on the precision ϵ of the HU routine. Additionally, we compare the HU approach to an alternative SDP relaxation technique, the *sums-of-squares* (SOS) method.

This chapter summarizes the findings of a collaborative study [Ost+25]. The implementation of the HU routine, along with the integration of the GW algorithm, was carried out by me. The formulation of the *affinity-based slotting problem* (ASP) and the *vehicle routing problem* (VRP), the conversion into QUBO instances, and the implementation of the SOS relaxation were performed by other contributors.

5.1 Sums-of-squares method

The sums-of-squares method offers an alternative approach to approximating a QUBO using an SDP relaxation. Unlike the GW algorithm, the SOS method provides only an upper bound on the objective value, without yielding a corresponding binary solution vector x. To derive the SOS relaxation, we reformulate the QUBO Equation (2.5) as

$$\begin{aligned} &\text{minimize} && \gamma \in \mathbb{R} \\ &\text{subject to} && \gamma - x^T C x \geq 0 \quad \text{for all } x \in \{-1, 1\}. \end{aligned} \tag{5.1}$$

With this formulation, every feasible γ in Equation (5.1) is an upper bound for the original QUBO (2.5). However, the main challenge now lies in enforcing the non-negativity constraint efficiently, as explicitly checking all possible values of x is computationally infeasible.

To approach this, we define the function

$$f(x) := \gamma - x^T C x. \tag{5.2}$$

To certify the nonnegativity of $f(x)$, we express it as a sum of squares of polynomials over x:

$$f(x) = \sum_i (s_i(x))^2 \quad \text{for all } x \in \mathbb{R}^n. \tag{5.3}$$

Since squares of real-valued polynomials are always nonnegative, this representation guarantees that $f(x) \geq 0$ for all $x \in \mathbb{R}$. To construct a suitable polynomial representation, we fix a maximum degree $2d$ of f and consider the vector

$$m_d = (1, x_1, \dots, x_n, x_1^2, x_1x_2, \dots) \in \mathbb{R}^{\binom{n+d}{d}} \tag{5.4}$$

containing all monomials up to degree d. Then, if $f(x)$ is a sum of squares, it can be written in the form

$$f(x) = (Bm_d)^T Bm_d = m_d^T B^T Bm_d \tag{5.5}$$

for some matrix B. Then, $G = B^T B$ is a psd matrix known as the *Gram matrix*. With this, we can reformulate (5.1) as the following SDP:

$$\begin{aligned} \text{minimize} \quad & \gamma \in \mathbb{R} \\ \text{subject to} \quad & \gamma - x^T Cx = m_d^T Gm_d, \\ & G \succeq 0. \end{aligned} \tag{5.6}$$

By increasing the degree d, we can achieve a more precise approximation at the cost of significantly increasing the dimension of the Gram matrix G. We refer to d as the *order* of the relaxation.

5.2 Problem formulations

Affinity-based slotting problem

In warehouse logistics, efficient item storage is crucial for optimizing retrieval processes, such as minimizing the number of aisle changes required when collecting a set of items. The *affinity-based slotting problem* (ASP) takes into account the fact that certain items are frequently ordered together, which is quantified by their *pairwise affinity*. The goal is to store items with high affinity close to one another to improve retrieval efficiency. This problem becomes even more challenging when considering storage capacity constraints, as each aisle has a limited capacity for different types of materials. The combination of these factors results in an NP-hard combinatorial optimization problem. In this work, we formulated ASP as an IQP (see Equation (2.4)) to capture both the affinity relationships and the capacity constraints effectively.

Vehicle routing problem

The *vehicle routing problem* (VRP) extends the well-known *traveling salesman problem* (TSP) by considering multiple vehicles instead of just one. The objective is

to determine optimal routes for a fleet of vehicles that must visit a set of customers while minimizing the total travel distance. Although the TSP is NP-hard, modern algorithms can solve many practical instances efficiently. However, TSP often oversimplifies real-world applications, as industrial logistics typically involve multiple vehicles and varying constraints on vehicle capacity and depot locations. VRPs introduce additional complexity by incorporating these real-world constraints, making them significantly harder to solve. In this work, we formulated the VRP as an ILP (see Equation (2.3)).

QUBO formulations

To apply the HU and SOS methods to the VRP and ASP, we must first reformulate them as (QUBO) problems. This transformation is well studied, and various approaches exist. For this work, we followed the methodology outlined by Glover et al. [GKD19]. The conversion to a QUBO consists of two parts:

- **Unconstraining and penalization:** The constraints in the IP formulation must be incorporated into the objective function. This is done by introducing *penalty terms*, ensuring that any constraint violation results in an increase in the objective value. Choosing appropriate penalty magnitudes is crucial. If they are too low, the optimal QUBO solution may violate constraints of the IP. On the other hand, if they are too high, the problem may become harder to solve, as the cost function could be dominated by the penalty terms.
- **Binarization:** Since a QUBO requires binary variables, integer values in the original IP formulation must be encoded using additional *slack variables*. The number of binary variables required scales logarithmically with the range of the integer variables, leading to an increase in the problem's dimension.

If penalty values are appropriately chosen, the optimal solution of the QUBO formulation is equivalent to that of the original IP problem. Consequently, any valid upper or lower bounds on the QUBO objective also apply to the original problem. However, approximate solutions obtained from the QUBO formulation do not necessarily correspond to feasible solutions for the original problem.

5.3 Benchmarks

5.3.1 Instances

For this thesis, we focus on one instance each of the VRP and ASP. The corresponding QUBO formulations have dimensions of $n = 96$ for the ASP and $n = 3380$ for the VRP. We denote their optimal objective values as γ_{opt}.

The IP formulations of both problems can be solved exactly using CPLEX [Cpl09]. However, the QUBO formulations showed to be harder to solve with standard solvers than the IP formulation. Here, only the QUBO solution for the smaller ASP instance could be obtained using Gurobi [Gur24], but not for the larger VRP instance. The corresponding running times and results are summarized in Table 5.1.

instance	dimension	optimal value γ_{opt}	time (IP) in s	time (QUBO) in s
ASP	96	246.78	0.45	< 10
VRP	3380	3036.54	1128.14	-

TABLE 5.1: Exact solutions of the ASP and VRP instance. Results were obtained using CPLEX for IP formulations and Gurobi for QUBO formulations on a CPU. Dimensions are given for the QUBO formulations.

5.3.2 Results

To evaluate the quality of the bounds computed using the SOS and HU methods, we compare them against the optimal solution of the original IP formulation. The absolute deviation of the computed lower or upper bound from the optimal solution γ_{opt} is given by

$$\Delta^{\text{abs}}_{\text{low/up}} = |\gamma_{\text{low/up}} - \gamma_{\text{opt}}|. \tag{5.7}$$

Since the absolute objective values depend on the specific weight choices and are not inherently meaningful, we consider the relative deviations. These are defined as

$$\Delta^{\text{rel}}_{\text{low/up}} = \frac{\Delta^{\text{abs}}_{\text{low/up}}}{\gamma_{\text{opt}}}. \tag{5.8}$$

The SOS relaxations were computed entirely on an Intel Core i7-8700 (CPU), while the HU routine was executed both on the same CPU and on an Nvidia GeForce RTX 4090 (GPU). As expected, GPU computations significantly outperformed CPU execution in terms of running time. To estimate the running time of the quantum implementation (QC) of the HU algorithm, we use a lower bound on the number of two-qubit gates derived in Section 4.2. Assuming a gate execution time of 6.5×10^{-9}s based on the current gate speed record [Che+22], we derive conservative running time estimates. The final results are presented in Tables 5.2 and 5.3.

	relative deviation of bounds		time in s		
	$\Delta^{\text{rel}}_{\text{up}}$	$\Delta^{\text{rel}}_{\text{low}}$	CPU	GPU	QC
$\text{HU}_{\epsilon=10^{-2}}$	945.6	16.8	-	0.17	8.4×10^{10}
$\text{HU}_{\epsilon=10^{-3}}$	86.0	14.2	-	3.08	1.2×10^{16}
$\text{HU}_{\epsilon=10^{-4}}$	8.8	6.6	-	500.36	3.1×10^{21}
$\text{HU}_{\epsilon=10^{-5}}$	3.8	3.2	-	21861.1	7.7×10^{25}
$\text{HU}_{\epsilon=10^{-5.5}}$	3.1	2.8	-	2.7×10^{5}	3.5×10^{28}
$\text{SOS}_{\text{1st order}}$	1.99	-	3.24	-	-
$\text{SOS}_{\text{2nd order}}$	0.13	-	28.46	-	-

TABLE 5.2: Results for solving the QUBO formulation of the ASP instance with dimension $n = 96$ using Hamiltonian Updates and sums-of-squares.

	relative deviation of bounds		time in s		
	$\Delta^{\text{rel}}_{\text{up}}$	$\Delta^{\text{rel}}_{\text{low}}$	CPU	GPU	QC
$\text{HU}_{\epsilon=10^{-2}}$	5.6×10^7	8.4×10^5	310.36	72.64	9.4×10^{12}
$\text{HU}_{\epsilon=10^{-3}}$	6.2×10^6	8.5×10^5	1314.95	266.42	2.0×10^{18}
$\text{HU}_{\epsilon=10^{-4}}$	5.5×10^5	2.2×10^4	-	37405.86	4.7×10^{24}
$\text{SOS}_{\text{1st order}}$	1.60	-	95.63	-	-

TABLE 5.3: Results for solving the QUBO formulation of the VRP instance with dimension $n = 3380$ using Hamiltonian Updates and sums-of-squares.

We find that the GW algorithm combined with the HU routine produces significantly looser upper bounds compared to the SOS method, with much higher relative deviations. For the smaller ASP instance, increasing the precision reduces this gap, but at the cost of an immensely higher computation time. In the larger VRP instance, the accuracy of the HU algorithm deteriorates even further, with relative errors increasing by several orders of magnitude.

Interestingly, the SOS method shows improved relative accuracy for the larger instance, at least in first-order relaxation. However, the increased problem dimension in the VRP caused solvers to fail when computing the second-order SOS relaxation. The estimated quantum running time for the HU algorithm is prohibitively high, exceeding 10^{10} seconds even for the simplest cases. Under the assumptions of this work, a quantum implementation of HU appears therefore entirely impractical.

To apply the HU algorithm, the cost matrix of the QUBO was normalized. After computation, results were converted back to the original formulation by reversing this rescaling. However, this also meant that the precision ϵ was effectively scaled by the operator norm of the cost matrix. This explains the significantly worse performance of the HU routine for the larger instance, where the norm is larger. Overall, our results suggest that approximate solvers like HU are not well-suited for QUBO formulations derived from integer programs. Further testing is necessary to determine whether HU could be effective for applications where QUBO formulations arise naturally rather than as a reformulation of an IP.

Chapter 6

Conclusion

In this work, we investigated whether the Hamiltonian Updates (HU) algorithm by [GLBKSF22] could achieve a practical speedup on future quantum hardware compared to classical implementations. To this end, we first improved the general performance of the HU routine by introducing a series of heuristics that reduced the algorithm's running time by a factor of over 40,000 for the tested instances. Additionally, we provided tighter theoretical bounds on the algorithm's scaling with precision, and numerically quantified its dependence.

By deriving lower bounds on the number of required quantum gate operations for the subroutines of HU, we benchmarked the algorithm on artificial data and systematically compared its classical and quantum performance. Our results show that even under highly favorable assumptions, the quantum implementation remains at least 10^7 times slower than its classical counterpart for realistically computable problem instances. In practical settings, where additional challenges such as error correction must be accounted for, this performance gap is likely to increase by several additional orders of magnitude.

In a second study, two real-world combinatorial optimization problems were formulated as integer programs (IPs) and converted into quadratic unconstrained binary optimization (QUBO) problems. We applied HU in combination with the Goemans-Williamson (GW) algorithm and compared its performance to a second semidefinite programming (SDP) relaxation approach, the sums-of-squares (SOS) method. Our findings reveal that even when using high precision for HU, the method produced extremely poor approximations in the original IP formulation. While the SOS approach provided tighter bounds than HU, it still performed significantly worse than conventional solvers. However, this does not necessarily imply that HU and SOS methods are fundamentally ineffective. Instead, our results suggest that QUBO based reformulations of IPs can be highly sensitive to solution precision. This observation is also relevant for other quantum optimization approaches such as QAOA and quantum annealing, which typically provide only approximate solutions and may struggle with such precision-related issues.

Our benchmarking results, demonstrating that the quantum implementation of HU is many orders of magnitude slower than its classical counterpart, are consistent with previous analyses of similar quantum optimization algorithms [Dal+23; Amm+23]. This suggests that a simple quadratic asymptotic speedup may often be insufficient

to achieve practical quantum advantage. Many proposed fault-tolerant quantum algorithms for optimization problems rely on techniques inspired by Grover's search, such as quantum minimum search [DH99] and quantum random walks [VA12]. These also provide, at best, a quadratic speedup. This aligns with the broader conjecture that NP-hard problems, where most challenging optimization tasks belong, cannot be efficiently solved on a quantum device. Ultimately, it remains questionable, whether quantum computers will ever be able to outperform classical computers for practical optimization problems.

Bibliography

[Aar05] Scott Aaronson. *NP-complete Problems and Physical Reality*. 2005. arXiv: quant-ph/0502072 [quant-ph].

[Ali95] Farid Alizadeh. "Interior point methods in semidefinite programming with applications to combinatorial optimization". In: *SIAM Journal on Optimization* (1995). DOI: 10.1137/0805002.

[Amb+] Andris Ambainis et al. "Quantum Speedups for Exponential-Time Dynamic Programming Algorithms". In: *Proceedings of the 2019 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*. DOI: 10.1137/1.9781611975482.107.

[Amm+23] Sabrina Ammann et al. *Realistic Runtime Analysis for Quantum Simplex Computation*. 2023. arXiv: 2311.09995 [quant-ph].

[AN06] Noga Alon and Assaf Naor. "Approximating the Cut-Norm via Grothendieck's Inequality". In: *SIAM Journal on Computing* (2006). DOI: 10.1137/S0097539704441629.

[Aru+19] Frank Arute et al. "Quantum supremacy using a programmable superconducting processor". In: *Nature* (2019). DOI: 10.1038/s41586-019-1666-5.

[Ben+97] Charles H. Bennett et al. "Strengths and Weaknesses of Quantum Computing". In: *SIAM Journal on Computing* (1997). DOI: 10.1137/S0097539796300933.

[Bev+22] Michael E. Beverland et al. *Assessing requirements to scale to practical quantum advantage*. 2022. arXiv: 2211.07629 [quant-ph].

[BJR89] F. Barahona, M. Jünger, and G. Reinelt. "Experiments in quadratic 0–1 programming". In: *Mathematical Programming* (1989). DOI: 10.1007/BF01587084.

[BM24] Sami Boulebnane and Ashley Montanaro. "Solving Boolean Satisfiability Problems With The Quantum Approximate Optimization Algorithm". In: *PRX Quantum* (3 2024). DOI: 10.1103/PRXQuantum.5.030348.

[Bol+24] Suresh Bolusani et al. *The SCIP Optimization Suite 9.0*. 2024. arXiv: 2402.17702 [math.OC].

[Bra+20] Sergey Bravyi et al. "Obstacles to Variational Quantum Optimization from Symmetry Protection". In: *Phys. Rev. Lett.* (26 2020). DOI: 10.1103/PhysRevLett.125.260505.

[BV94] Stephen Boyd and Lieven Vandenberghe. *Linear matrix inequalities in system and control theory*. SIAM Studies in Applied Mathematics. SIAM, 1994.

[Che+22] Y. Chew et al. "Ultrafast energy exchange between two single Rydberg atoms on a nanosecond timescale". In: *Nature Photonics* (2022). DOI: 10.1038/s41566-022-01047-2.

[CL24] François Clautiaux and Ivana Ljubić. "Last fifty years of integer linear programming: A focus on recent practical advances". In: *European Journal of Operational Research* (2024). DOI: https://doi.org/10.1016/j.ejor.2024.11.018.

[Cpl09] IBM ILOG Cplex. "V12. 1: User's Manual for CPLEX". In: *International Business Machines Corporation* (2009).

[Dal+23] Alexander M. Dalzell et al. "End-To-End Resource Analysis for Quantum Interior-Point Methods and Portfolio Optimization". In: *PRX Quantum* (4 2023). DOI: 10.1103/PRXQuantum.4.040325.

[DH99] Christoph Durr and Peter Hoyer. *A Quantum Algorithm for Finding the Minimum*. 1999. arXiv: quant-ph/9607014 [quant-ph].

[DP+23] Giacomo De Palma et al. "Limitations of Variational Quantum Algorithms: A Quantum Optimal Transport Approach". In: *PRX Quantum* (1 2023). DOI: 10.1103/PRXQuantum.4.010309.

[Drm+25] P. Drmota et al. "Experimental Quantum Advantage in the Odd-Cycle Game". In: *Phys. Rev. Lett.* (7 2025). DOI: 10.1103/PhysRevLett.134.070201.

[DSGW18] Himeshi De Silva, John L. Gustafson, and Weng-Fai Wong. "Making Strassen Matrix Multiplication Safe". In: *2018 IEEE 25th International Conference on High Performance Computing (HiPC)*. 2018. DOI: 10.1109/HiPC.2018.00028.

[Far+01] Edward Farhi et al. "Quantum computation by adiabatic evolution". In: *arXiv preprint quant-ph/0001106* (2001). arXiv: quant-ph/0001106.

[Far+22] Edward Farhi et al. "The Quantum Approximate Optimization Algorithm and the Sherrington-Kirkpatrick Model at Infinite Size". In: *Quantum* (2022). DOI: 10.22331/q-2022-07-07-759.

[FGG14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. *A Quantum Approximate Optimization Algorithm*. 2014. arXiv: 1411.4028 [quant-ph].

[FGG20] Edward Farhi, David Gamarnik, and Sam Gutmann. *The Quantum Approximate Optimization Algorithm Needs to See the Whole Graph: Worst Case Examples*. 2020. arXiv: 2005.08747 [quant-ph].

[GJ79] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[GKD19] Fred Glover, Gary Kochenberger, and Yu Du. *A Tutorial on Formulating and Using QUBO Models*. 2019. arXiv: 1811.11538 [cs.DS].

[GLBKSF22] Fernando G.S L. Brandão, Richard Kueng, and Daniel Stilck França. "Faster quantum and classical SDP approximations for

quadratic binary optimization". In: *Quantum* (2022). DOI: 10.22331/q-2022-01-20-625.

[Got98] Daniel Gottesman. "Theory of fault-tolerant quantum computation". In: *Phys. Rev. A* (1 1998). DOI: 10.1103/PhysRevA.57.127.

[Gro96] Lov 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. Philadelphia, Pennsylvania, USA: Association for Computing Machinery, 1996. DOI: 10.1145/237814.237866.

[Gur24] Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2024.

[GW95] Michel X. Goemans and David P. Williamson. "Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming". In: *Journal of the ACM (JACM)* (1995). DOI: 10.1145/227683.227684.

[Hen+25] Fabian Henze et al. *Solving quadratic binary optimization problems using quantum SDP methods: Non-asymptotic running time analysis*. 2025. arXiv: 2502.15426 [quant-ph].

[Kar72] Richard M. Karp. "Reducibility among Combinatorial Problems". In: *Complexity of Computer Computations*. 1972. DOI: 10.1007/978-1-4684-2001-2_9.

[Kar84] Narendra Karmarkar. "A new polynomial-time algorithm for linear programming". In: *Combinatorica* (1984). DOI: 10.1007/BF02579150.

[Kho+07] Subhash Khot et al. "Optimal Inapproximability Results for MAX-CUT and Other 2-Variable CSPs?" In: *SIAM Journal on Computing* (2007). DOI: 10.1137/S0097539705447372.

[KM72] Victor Klee and George J. Minty. "How good is the simplex algorithm?" In: *Inequalities III*. Academic Press, 1972.

[Lan+04] Gert R. Lanckriet et al. "Learning the kernel matrix with semidefinite programming". In: *Journal of Machine Learning Research* (2004).

[Luc14] Andrew Lucas. "Ising formulations of many NP problems". In: *Frontiers in Physics* (2014). DOI: 10.3389/fphy.2014.00005.

[McC+16] Jarrod R. McClean et al. "The theory of variational hybrid quantum-classical algorithms". In: *New Journal of Physics* (2016). DOI: 10.1088/1367-2630/18/2/023023.

[Mon20] Ashley Montanaro. "Quantum speedup of branch-and-bound algorithms". In: *Phys. Rev. Res.* (1 2020). DOI: 10.1103/PhysRevResearch.2.013056.

[NN94] Yurii Nesterov and Arkadii Nemirovski. *Interior-point polynomial algorithms in convex programming*. SIAM, 1994.

[Ost+25] Birte Ostermann et al. *Benchmarking of quantum and classical SDP relaxations for QUBO formulations of real-world logistics problems*. 2025. arXiv: 2503.10801 [math.OC].

[Per+14] Alberto Peruzzo et al. "A variational eigenvalue solver on a photonic quantum processor". In: *Nature Communications* (2014). DOI: 10.1038/ncomms5213.

[Phi+22] Stephan G. J. Philips et al. "Universal control of a six-qubit quantum processor in silicon". In: *Nature* (2022). DOI: 10.1038/s41586-022-05117-x.

[PKG24] Sieglinde M.-L. Pfaendler, Konstantin Konson, and Franziska Greinert. "Advancements in Quantum Computing—Viewpoint: Building Adoption and Competency in Industry". In: *Datenbank-Spektrum* (2024). DOI: 10.1007/s13222-024-00467-4.

[Pre18] John Preskill. "Quantum Computing in the NISQ era and beyond". In: *Quantum* (2018). DOI: 10.22331/q-2018-08-06-79.

[Res19] IBM Research. *On "Quantum Supremacy"*. IBM Research Blog. 2019.

[Sho97] Peter W. Shor. "Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer". In: *SIAM Journal on Computing* (1997). DOI: 10.1137/S0097539795293172.

[SK22] Maria Schuld and Nathan Killoran. "Is Quantum Advantage the Right Goal for Quantum Machine Learning?" In: *PRX Quantum* (3 2022). DOI: 10.1103/PRXQuantum.3.030101.

[Str69] Volker Strassen. "Gaussian elimination is not optimal". In: *Numerische Mathematik* (1969). DOI: 10.1007/BF02165411.

[Tod01] M. J. Todd. "Semidefinite optimization". In: *Acta Numerica* (2001). DOI: 10.1017/S0962492901000071.

[VA12] Salvador Elías Venegas-Andraca. "Quantum walks: a comprehensive review". In: *Quantum Information Processing* (2012). DOI: 10.1007/s11128-012-0432-5.

[Wol20] Laurence A Wolsey. *Integer programming*. John Wiley & Sons, 2020.

[Xue+22] Xiao Xue et al. "Quantum logic with spin qubits crossing the surface code threshold". In: *Nature* (2022). DOI: 10.1038/s41586-021-04273-w.

[Zho+21] Han-Sen Zhong et al. "Phase-Programmable Gaussian Boson Sampling Using Stimulated Squeezed Light". In: *Phys. Rev. Lett.* (18 2021). DOI: 10.1103/PhysRevLett.127.180502.