A COPOSITIVE FRAMEWORK FOR ANALYSIS OF HYBRID ISING-CLASSICAL ALGORITHMS

ROBIN BROWN[*†‡] , DAVID E. BERNAL NEIRA[†‡§] , DAVIDE VENTURELLI[†‡] , AND MARCO PAVONE[*]

Abstract. Recent years have seen significant advances in quantum/quantum-inspired technologies capable of approximately searching for the ground state of Ising spin Hamiltonians. The promise of leveraging such technologies to accelerate the solution of difficult optimization problems has spurred an increased interest in exploring methods to integrate Ising problems as part of their solution process, with existing approaches ranging from direct transcription to hybrid quantum-classical approaches rooted in existing optimization algorithms. While it is widely acknowledged that quantum computers should augment classical computers, rather than replace them entirely, comparatively little attention has been directed toward deriving analytical characterizations of their interactions. In this paper, we present a formal analysis of hybrid algorithms in the context of solving mixed-binary quadratic programs (MBQP) via Ising solvers. By leveraging an existing completely positive reformulation of MBQPs, as well as a new strong-duality result, we show the exactness of the dual problem over the cone of copositive matrices, thus allowing the resulting reformulation to inherit the straightforward analysis of convex optimization. We propose to solve this reformulation with a hybrid quantum-classical cutting-plane algorithm. Using existing complexity results for convex cutting-plane algorithms, we deduce that the classical portion of this hybrid framework is guaranteed to be polynomial time. This suggests that when applied to NP-hard problems, the complexity of the solution is shifted onto the subroutine handled by the Ising solver.

1. Introduction.

Recent years have seen significant advances in quantum and quantum-inspired Ising solvers, such as quantum annealers [1], quantum approximate optimization circuits [2], or coherent Ising machines [3]. These are devices/methodologies designed to heuristically compute solutions of optimization problems of the form: $\min_{z\in\{-1,1\}^n}\sum_{i,j} J_{i,j}z_iz_j+\sum_i h_iz_i$, where $J_{i,j}$, h_i are real coefficients and $z_i \in \{-1,1\}$ are discrete variables to be optimized over. The promise of leveraging such technologies to speed up the solution of complex optimization problems has spurred many researchers to explore how Ising solvers can be applied to problems in various domains.

A standard approach has emerged where an optimization problem is directly transcribed into an Ising problem, and the returned solution is taken at face value or with minimal post-processing. While this method works well for problems that organically have an Ising form, sequences of reformulations can result in ill-conditioning of the problem in terms of the number of additional variables, the coupling strengths, and the optimization landscape. Most unnaturally, however, relying solely on the Ising solver means forgoing the advantages of already powerful classical computers.

In an effort to introduce meaningful interplay between classical and quantum computers, a few authors have proposed decomposition methods based on the Alternating Direction Method of Multipliers (ADMM), [4], or Benders Decomposition (BD), [5, 6, 7]. Critically, when ADMM is applied to non-convex problems, it is not guaranteed to converge. When it does, it is often to a local optimum without convergence guarantees to global optimality. On the other hand, while it may be possible to derive optimality guarantees using BD, proving convergence typically relies on an exhaustive search through the “complicating variables”. This makes it unclear whether such an algorithmic scaffold is primed to take advantage of speed-ups that the Ising solver may offer.

Contributions. Our work is motivated by the desire to rigorously analyze the interplay between classical and quantum machines in hybrid algorithms. Such analysis is a cornerstone for articulating and setting standards for hybrid quantum-classical optimization algorithms. Specifically, we espouse convergent hybrid quantum-classical algorithms that (1) use Ising solvers as a primitive while offering some resilience to their heuristic nature and (2) have polynomial complexity in the classical portions of the algorithm. To this end, the contribution of this paper is an algorithmic framework that satisfies these key desiderata. Concretely,

1. We revisit and prove strong duality of a result in [8] to show that the convex copositive formulation of many mixed-binary quadratic optimization problems is exact. Neglecting the challenges of working with copositive matrices, convex programs are a well-understood class of optimization problems with a wide variety of efficient solution algorithms. By reformulating mixed-binary quadratic programs as copositive programs, we open the door for hybrid-quantum classical algorithms that are based on existing convex optimization algorithms.
2. To solve the copositive programs, we propose applying a standard cutting-plane algorithm,

[*] Stanford University, Autonomous Systems Laboratory
[†] USRA Research Institute for Advanced Computer Science (RIACS)
[‡] NASA Quantum Artificial Intelligence Laboratory (QuAIL)
[§] Purdue University, Davidson School of Chemical Engineering

which we modify in a novel way using a hybrid quantum-classical approach. In particular, the cutting-plane algorithm serves as a template for a solution algorithm that alternates between checking copositivity and other operations. We hybridize the algorithm by approximating the copositivity checks via discretization and solving them with an Ising solver. We show that the complexity of the portion of the algorithm handled by the classical computer has polynomial scaling. This analysis suggests that when applied to NP-hard problems, the complexity of the solution is shifted onto the subroutine handled by the Ising solver.

3. We conducted benchmarking based on the maximum clique problem to validate our theoretical claims and evaluate potential speed-ups from using a stochastic Ising solver in lieu of a state-of-the-art deterministic solver or an Ising heuristic. Results indicate that the Ising formulation of the subproblems of the hybrid algorithm is efficient versus a MIP formulation in `Gurobi`, and the hybrid algorithm potentially is competitive even against a non-hybridized Ising formulation of the full problem solved by simulated annealing.

We emphasize that the contribution of this work is not the copositive reformulation or a novel cutting-plane algorithm but rather the insight that these ideas are synergistic with recent advances in quantum(-inspired) computing. In particular, copositive optimization is useful for deriving and analyzing new hybrid algorithms rooted in existing convex optimization algorithms, thus filling a gap in the hybrid algorithms literature.

While preparing this manuscript, a hybrid quantum-classical method relying upon a Frank-Wolfe method was published [9]. This work also leverages a similar copositive reformulation of quadratic binary optimization problems. We highlight the differences between that work and the one in our manuscript below. This manuscript considers the optimization problem class of mixed-binary quadratic programs, while in [9], the authors propose their method for quadratic binary optimization problems, a subcase of the problems considered herein. Moreover, we provide proof of the exactness and strong duality for copositive/completely positive optimization stemming from the mixed-integer quadratic reformulation, addressing an open question in the field. In their manuscript, [9] conjectures the results proved in this manuscript to be true. Finally, our solution method, which is based on cutting-plane algorithms, has a potential exponential speed-up in runtime compared to Frank-Wolfe algorithms.

1.1. Related Work. One dominant method for mapping optimization problems into Ising problems is through direct transcription. This process typically involves discretizing continuous variables and passing constraints into the objective through a penalty function; the returned solution is often taken at face value or with minimal post-processing to enforce feasibility. Owing to its simplicity, this process has found applications in a variety of problems, including jobshop scheduling [10], routing problems [11], community detection [12], and all of Karp's 21 NP-complete models [13], among others. Critically, unless a problem organically takes an Ising form, this approach often requires many auxiliary variables (spins), introduces large skews in the coupling coefficients, and can result in poor conditioning of the optimization landscape, thus limiting the problems that can be solved on near-term devices. Similarly, extending the class of applicable problem instances requires deriving increasingly complex sequences of reformulations, each of which reduces the solubility of the final reformulation. More importantly, an algorithm with minimal interactions between classical and quantum computers disregards the bountiful successes of classical computers in the past decades. This has inspired some researchers to examine how quantum computers can be used to augment classical computers instead of replacing them entirely [14].

As an alternative to direct transcription, there is a burgeoning body of literature exploring the potential of decomposition methods for designing hybrid quantum-classical algorithms. These generally refer to algorithms that divide effort between a classical and quantum computer, with each computer informing the computation carried out by the other. Among these, algorithms based on the Benders Decomposition (BD) are gaining traction. BD is particularly effective for problems characterized by "complicating variables", for which the problem becomes easy once these variables are fixed. For example, a mixed-integer linear program (MILP) becomes a linear program (LP) once the integer variables are fixed–the integers are the complicating variables. BD iterates between solving a master problem over the complicating variables and sub-problems where the complicating variables are fixed, whose solution is used to generate cuts for the master problem. Both [5] and [6] consider mixed-integer programming (MIP) problems where the integer variables are linked to the continuous variables through a polyhedral constraint and leverage a reformulation where dependence on the continuous variables is expressed as constraints over the extreme rays and points of the original feasible region. Because the number of extreme rays and points may be exponentially large, the constraints are not written down in full but are iteratively generated from the solutions of the sub-problems. The master problem is an integer program consisting of these constraints and is solved using the quantum computer. Notably, the generated constraint set may be large, with the worst case being the generation of the entire constraint set, resulting in a large number of iterations. The approach in [7] attempts to mitigate this by generating multiple cuts per iteration and selecting the most informative subset of these cuts. Instead of using the quantum computer to solve the master problem, the quantum computer is used to heuristically select cuts based on a minimum set cover or maximum coverage metric. While this may effectively reduce the number of iterations and size of the constraint set, the master problem is often an integer program that may be computationally intractable. For each of the proposed approaches, it is unclear how the complexity of the problem is distributed through the solution process For example, for [5, 6] the complexity might show up in the number of iterations, and for [7] it might show up when solving the master problem. Consequently, it is ambiguous whether BD-based approaches can take advantage of a speed-up in the Ising solver, even if one were to exist.

Another decomposition that has been explored is based on the Alternating Direction Method of Multipliers (ADMM)[4]. This is an algorithm to decompose large-scale optimization problems into smaller, more manageable sub-problems [15]. While originally designed for convex optimization, ADMM has shown great success as a heuristic for non-convex optimization as well, [16], and significant progress has been made towards explaining its success in such settings [17]. In [4], the authors propose an ADMM-based decomposition with three sub-problems: the first being over just the binary variables, the second being the full problem with a relaxed copy of the binary variables, and the third being a term that ties the binary variables and their relaxed copies together. For quadratic pure-binary problems, the authors show that the algorithm converges to a stationary point of the augmented Lagrangian, which may not be a global optimizer Convergence to a global optimum is only guaranteed under the more stringent Kurdyka-Łojasiewicz conditions on the objective function [18]. Unfortunately, the assumptions guaranteeing convergence to a stationary point fail in the presence of continuous variables.

A third class of decomposition proposed and implemented in the `qbsolv` solver is based on tabu search [19]. `qbsolv` can be seen as iterating between a large-neighborhood local search (using an Ising solver) and tabu improvements to locally refine the solution (using a classical computer), where previously found solutions are removed from the search space in each iteration. During the local search phase, subsets of the variables are jointly optimized while the remaining variables are fixed to their current values. The solution found in this phase is then used to initialize the tabu search algorithm, and the process is repeated for a fixed number of iterations. Critically, it is unclear whether the algorithm is guaranteed to converge and, if so, what its optimality guarantees are. While finite convergence of tabu search is investigated in [20], it relies on either recency or frequency memory that ensures an exhaustive search of all potential solutions.

Another approach for purely integer programming problems is based on the computation of a Graver basis through the computation of the integer null-space of the constraint set as proposed in [21]. This null-space computation is posed as a quadratic unconstrained binary optimization (QUBO) and then post-processed to obtain the Graver basis of the constraint set, a test-set of the problem. The test-set provides search directions for an augmentation-based algorithm. For a convex objective, it provides a polynomial oracle complexity in converging to the optimal solution. The authors initialize the problem by solving a feasibility-based QUBO and extend this method to non-convex objectives by allowing multiple starting points for the augmentation. The multistart procedure also alleviates the requirement for computing the complete Graver basis of the problem, which grows exponentially with the problem's size. Considering an incomplete basis or non-convex objectives makes the Graver Augmentation Multistart Algorithm (GAMA) a heuristic for general integer programming problems, and it cannot address problems with continuous variables.

In this paper, we seek to address a gap in the literature on a rigorous theory of hybrid quantum-classical optimization. By revisiting the hidden convex structure of non-convex problems, we pave the way for hybrid algorithms based on efficient convex optimization. We show that algorithms derived through this approach inherit the straightforward analysis of convex optimization without sacrificing the potential benefits of quantum computing for non-convex problems.

While there is optimism regarding improvements to and our understanding of quantum technology in the coming decades, few expect that they will replace classical computers entirely. We believe that the method presented in this paper is an approach to algorithm design that anticipates a future where quantum computers and classical computers work in tandem. In particular, we envision a mature theory of hybrid algorithms that clearly delineates how quantum and classical computers should complement each other.

1.2. Quantum/Quantum-inspired Ising Solvers. Adiabatic quantum computing (AQC) is a quantum computation paradigm that operates by initializing a system in the ground state of an initial Hamiltonian (i.e., the optimal solution of the corresponding objective function) and slowly sweeping the system to an objective Hamiltonian. This Hamiltonian, referred to as the cost Hamiltonian, maps the objective function of the classical Ising model onto a system with as many quantum bits, or qubits, as original variables in the Ising model. The adiabatic theorem of quantum mechanics states that if the system evolution is "sufficiently slow", the system ends up in the ground state of the desired Hamiltonian. Here, "sufficiently slow" depends on the minimum energy gap between the ground and the first excited state throughout the system evolution [22]. Since the evaluation of the minimal gap is mostly intractable, one is forced to phenomenologically "guess" the evolution's speed, and if it is too fast, the undesired non-adiabatic transitions can occur. Additionally, real devices are plagued with various incarnations of physical noise, such as thermal fluctuations or decoherence effects, that can hamper computation. The situation is further exacerbated by the challenge of achieving dense connectivity between qubits. Densely connected problems are embedded in devices by chaining together multiple physical qubits to represent one logical qubit. The heuristic computational paradigm that encompasses the additional noise and non-quantum effects is known as Quantum Annealing (QA). [23] provides a review on QA with a focus on possible routes towards solving the open questions in the field.

An alternative paradigm to AQC is the gate-based model of quantum computing. Within the gate-based model, Variational Quantum Algorithms (VQAs) is a class of hybrid quantum-classical algorithms that can be applied to optimization [24]. VQAs share a common operational principle where the "loss function" of a parameterized quantum circuit is measured on a quantum device and evaluated on a classical processor, and a classical optimizer is used to update (or "train") the circuit's parameters to minimize the loss. VQAs are often interpreted as a quantum analog to machine learning, leaving many similar questions open regarding their trainability, accuracy, and efficiency. Most similar in spirit to this work is a theory of variational hybrid quantum-classical algorithms proposed in [25]. However, they primarily focus on algorithmic improvements to the quantum portion, with discussion of the classical optimization being limited to empirical evaluations of existing derivative-free optimization algorithms. More recently, [26] analyzed the complexity of training VQAs, and through reductions from the maximum cut problem, showed that it is NP-hard. The analysis presented in this paper complements these prior works in developing a more complete picture of the interplay between quantum and classical computers.

The quantum approximate optimization algorithm (QAOA) is a specific instance of a VQA where the structure of the quantum circuit is the digital analog of adiabatic quantum computing [2]. QAOA operates by alternating the application of the cost Hamiltonian and a mixing Hamiltonian; the number of alternating blocks is referred to as the circuit depth. For each one of the alternating steps, either mixing or cost application, a classical optimizer needs to determine how long each step should be performed, encoded as rotation angles. Optimizing the expected cost function with respect to the rotation angles is a continuous low-dimensional non-convex problem. QAOA is designed to optimize cost Hamiltonians, such as the ones derived from classical Ising problems. Performance guarantees can be derived for QAOA with well-structured problems, given that the optimal angles are found in the classical optimization step. Although approximation guarantees have not been derived for arbitrary cost Hamiltonians, even depth-one QAOA circuits have non-trivial performance guarantees for specific problems and cannot be efficiently simulated on classical computers [27], thus bolstering the hope for a speed-up in near-term quantum machines. Moreover, the algorithm's characteristics, such as relatively shallow circuits, make it amenable to be implemented in currently available noisy intermediate-scale quantum (NISQ) computers compared to other algorithms requiring fault-tolerant quantum devices [28]. While QAOA's convergence to optimal solutions is known to improve with increased circuit depth and to succeed in the infinite depth limit following its equivalence to AQC, its finite depth behavior has remained elusive due to the challenges in analyzing quantum many-body dynamics and other practical complications such as decoherence when implementing long quantum circuits, compilation issues, and hardness of the optimal angle classical problem [29]. Even considering these complications, QAOA has been extensively studied and implemented in current devices [30, 31], becoming one of the most popular alternatives to address combinatorial optimization problems modeled as Ising problems using gate-based quantum computers. Several other quantum heuristics for Ising problems have been proposed, usually requiring fault-tolerant quantum computers. We direct the interested reader to a recent review on the topic [32].

An alternative physical system for solving Ising problems that has emerged is coherent Ising machines (CIMs), which are optically pumped networks of coupled degenerate optical parametric oscillators. As the pump strength increases, the equilibrium states of an ideal CIM correspond to the Ising Hamiltonian's ground states encoded by the coupling coefficients. Large-scale prototypes of CIMs have achieved impressive performance in the lab, thus driving the theoretical study of their fundamental operating principles. While significant advances have been made on this front, we still lack a clear theoretical understanding of the CIMs' computational performance. Since a thorough understanding of the CIM is limited by our capacity to prove theorems about complex dynamic systems, near-term usage of CIMs must treat them as a heuristic rather than a device with performance guarantees [33]. Even so, there are empirical observations that in many cases, the median complexity of solving Ising problems using CIM scales as $\exp \sqrt{N}$ where N is the size of the problem [34], making it a potential approach to solve these problems efficiently in practice. We note that there are other types of Ising machines, including classical thermal annealers (based on magnetic devices [35], optics [36], memristors [37], and digital hardware accelerators [38]), dynamical-systems solvers (based on optics [39] and electronics [40]), superconducting-circuit quantum annealers [41], and neutral atoms arrays [42]. We direct the interested reader to [34], which provides a recent review and comparison of various methods for constructing Ising machines and their operating principles.

Organization. In Section 2, we present notation, terminology, and the problem setting covered by our approach. In Section 3, we introduce the proposed framework, including convex reformulation via copositive programming, a high-level overview of cutting-plane algorithms, and a specific discussion of their application to copositive programming. Section 4 provides numerical experiments supporting our assertions about the proposed approach. Finally, we conclude and highlight future directions in Section 5.

2. Preliminaries.

2.1. Notation and Terminology. In this paper, we solely work with vectors and matrices defined over the real numbers and reserve lowercase letters for vectors and uppercase letters for matrices. We will also follow the convention that a vector $x \in \mathbb{R}^n$ is to be treated as a column vector, i.e., equivalent to a matrix of dimension $n \times 1$. For a matrix M, we use $M_{i,j}$ to denote the entry in the ith row and jth column, $M_{i,*}$ denotes the entire ith row, and $M_{*,j}$ denotes the entire jth column. In the text, we frequently use block matrices with structured zero entries; we use $\cdot$ as shorthand for zero entries. We use $\mathbb{1}$ to denote the all-ones vectors and $\mathbb{1}_{\{j\}}$ to denote the jth standard basis vector (i.e., a vector where all entries are zero except for a 1 for the jth entry). The p-norm of a vector $v \in \mathbb{R}^n$ is defined as $\|v\|_p := \left(\sum_{i=1}^n v_i^p\right)^{1/p}$. We reserve the letter I to denote the identity matrix. For two matrices, M and N, we use $\langle M, N\rangle = \mathrm{Tr}(M^\top N)$ to denote the matrix inner product. Note that for two vectors, $\mathrm{Tr}(x^\top y) = x^\top y$ because $x^\top y$ is a scalar, so the matrix inner product is consistent with the standard inner product on vectors. For sets, $S_M + S_N := \{M + N \mid M \in S_M, N \in S_N\}$ is their Minkowski sum, $S_M \cup S_N$ their union, and $S_M \cap S_N$ their intersection. For a cone, $\mathcal{K}$, its dual cone is defined as $\mathcal{K}^* = \{X \mid \langle X, K\rangle \geq 0, \forall K \in \mathcal{K}\}$. While we work with matrix cones in this paper, this definition of dual cones is consistent with vector cones as well. In this paper, the two cones we will work with are the cone of completely positive matrices and the cone of copositive matrices. The cone of completely positive (CP) matrices, C^*, is the set of matrices that have a factorization with entry-wise non-negative entries:

$$\mathcal{C}_n^* := \{X \in \mathbb{R}^{n\times n} \mid X = \sum_k x^{(k)}(x^{(k)})^\top, \quad x^{(k)} \in \mathbb{R}^n_{\geq 0}\} \tag{2.1}$$

The cone of copositive matrices, $\mathcal{C}$, is the set of matrices defined by:

$$\mathcal{C}_n := \{X \in \mathbb{R}^{n\times n} \mid v^\top X v \geq 0, \quad \forall v \in \mathbb{R}^n_{\geq 0}\} \tag{2.2}$$

As suggested by the notation, the cones of completely positive and copositive matrices are duals of each other. We use S^n_{++} to denote the cone of positive definite matrices.

In this paper, we will use the terms Ising problem and quadratic unconstrained binary optimization (QUBO) interchangeably. An Ising problem is an optimization problem of the form: $\min_{z\in\{-1,1\}^n} \sum_{i,j} J_{i,j} z_i z_j + \sum_i h_i z_i$, where $J_{i,j}$, h_i are real coefficients and $z_i \in \{-1, 1\}$ are discrete variables to be optimized over. A QUBO, which is an optimization problem of the form $\min_{x\in\{0,1\}^n} \sum_{i,j} Q_{i,j} x_i x_j$ can be reformulated as an Ising problem using the change of variable $z = 2x - \mathbb{1}$. This translates to coefficients in the Ising problem $J_{i,j} = \frac{1}{4} Q_{i,j}$, $h_i = \frac{1}{2}\sum_j Q_{i,j}$, and a constant offset of $\frac{1}{4}\sum_{i,j} Q_{i,j}$.

2.2. Problem Setting. In this paper, we consider mixed-binary quadratic programs (MBQP) of the form:

$$
\text{(MBQP)} \qquad \begin{array}{cl}
\underset{x \in \mathbb{R}^n}{\text{minimize}} & x^\top Q x + 2c^\top x \\
\text{subject to} & Ax = b, \quad A \in \mathbb{R}^{m \times n},\, b \in \mathbb{R}^m, \\
& x \geq 0, \\
& x_j \in \{0,1\}, \quad j \in B
\end{array}
$$

where the set $B \subseteq \{1, \dots, n\}$ indexes which of the n variables are binary. This is a general class of problems that encompasses problems including QUBOs, standard quadratic programming, the maximum stable set problem, and the quadratic assignment problem. Because mapping to an Ising problem can also be equivalently expressed as a QUBO, many problems tackled with Ising solvers thus far pass through a formulation similar to the form of Problem (MBQP). Using the result in [8, Sec. 3.2], the formulation considered in this paper can be extended to include constraints of the form $x_i x_j = 0$ that force at least one of x_i or x_j to be zero, i.e., complementarity constraints. For ease of notation, this extension is left out of the present discussion.

3. Proposed Methodology. In this section, we will discuss our proposed methodology for solving Problem (MBQP) given access to Ising solvers. Our result relies on a convex reformulation of Problem (MBQP) as a copositive program. Leveraging convexity, we propose to solve the problem using cutting-plane algorithms. These belong to a broad class of convex optimization algorithms whose standard components give rise to a natural separation between the role of the Ising solver versus a classical computer.

We first state Burer's exact reformulation of Problem (MBQP) as a completely positive program and its dual copositive program. We then show that under mild conditions (i.e., feasibility and boundedness) of the original MBQP, the copositive and completely positive programs exhibit strong duality. We will then introduce the class of cutting-plane algorithms and summarize the complexity guarantees of several well-known variants. Finally, we explicitly show how cutting-plane algorithms can be used to solve copositive optimization problems given a copositivity oracle and discuss how to implement a copositivity oracle using an Ising solver.

3.1. Convex formulation as a copositive program. In his seminal work, Burer showed that MBQPs can be represented exactly as completely positive programs of the form:

$$
\text{(CPP)} \qquad \begin{array}{cl}
\underset{X \in \mathbb{R}^{n \times n},\, x \in \mathbb{R}^n}{\text{minimize}} & \left\langle \begin{pmatrix} Q & c \\ c^\top & \cdot \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle \\
\text{subject to} & \left\langle \begin{pmatrix} \cdot & \frac{1}{2} A_{i,*}^\top \\ \frac{1}{2} A_{i,*} & \cdot \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle = b_i,\ i = 1, \dots, m, \\
& \left\langle \begin{pmatrix} A_{i,*}^\top A_{i,*} & \cdot \\ \cdot & \cdot \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle = b_i^2,\ i = 1, \dots, m, \\
& \left\langle \begin{pmatrix} -\mathbb{1}_{\{j\}} \mathbb{1}_{\{j\}}^\top & \frac{1}{2} \mathbb{1}_{\{j\}} \\ \frac{1}{2} \mathbb{1}_{\{j\}}^\top & \cdot \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle = 0,\ j \in B, \\
& \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \in \mathcal{C}_{n+1}^*,
\end{array}
$$

where exactness means that Problems (MBQP) and (CPP) have the same optimal objective and for an optimal solution, (x^*, X^*), of (CPP), x^* lies within the convex hull of optimal solutions for (MBQP) [8, Theorem 2.6]. Similar to semi-definite programming (SDP) relaxations, the completely positive formulation involves lifting the variables in (MBQP) to a matrix variable representing their first and second-degree monomials, making the objective function and constraints linear. Unlike SDP relaxations, however, the complete positivity constraint is sufficient for ensuring that the feasible region of (CPP) is exactly the convex hull of the feasible region of (MBQP). This distinction is what ensures that the optimal value of (CPP) is exactly that of (MBQP), whereas for an SDP relaxation, the optimal solution may lie outside of the convex hull of (MBQP), resulting in a lower objective value (i.e., a *relaxation gap*).

Taking the dual of (CPP) yields a copositive optimization problem of the form [43, Section 5.9]:

$$
\text{(COP)} \qquad \begin{array}{cl}
\underset{\mu, \lambda, \gamma}{\text{maximize}} & \gamma + \sum_{i=1}^{m} \mu_i^{(\text{lin})} b_i + \mu_i^{(\text{quad})} b_i^2 \\
\text{subject to} & M(\mu, \lambda, \gamma) \in \mathcal{C}_{n+1},
\end{array}
$$

where

$$
\begin{aligned}
M(\mu,\lambda,\gamma) :=& \begin{pmatrix} Q & c \\ c^\top & \cdot \end{pmatrix} - \sum_{i=1}^{m} \mu_i^{(\mathrm{lin})} \begin{pmatrix} \cdot & \frac{1}{2}A_{i,*}^\top \\ \frac{1}{2}A_{i,*} & \cdot \end{pmatrix} - \sum_{i=1}^{m} \mu_i^{(\mathrm{quad})} \begin{pmatrix} A_{i,*}^\top A_{i,*} & \cdot \\ \cdot & \cdot \end{pmatrix} \\
& - \sum_{j\in B} \lambda_j \begin{pmatrix} -\mathbb{1}_{\{j\}}\mathbb{1}_{\{j\}}^\top & \frac{1}{2}\mathbb{1}_{\{j\}} \\ \frac{1}{2}\mathbb{1}_{\{j\}}^\top & \cdot \end{pmatrix} - \gamma \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix}
\end{aligned}
\tag{3.1}
$$

is a parametrized linear combination of the constraint matrices. The dual copositive program has a linear objective and a single copositivity constraint–this is a convex optimization problem. While weak duality always holds between an optimization problem and its dual, strong duality is not generally guaranteed. Showing that strong duality holds is critical for ensuring convergence of specific optimization algorithms and exactness when solving the dual problem as an alternative to solving the primal.

Theorem 3.1 (Strong Duality). *If Problem (MBQP) is feasible with bounded feasible region, then strong duality holds between Problems (CPP) and (COP) (i.e.,* $\min$ (CPP) $=$ $\max$ (COP)*).*

Proof Sketch. Our proof proceeds by first showing strong duality between the alternative representation of (CPP) (using a homogenized formulation of the equality constraints) and its dual. By showing that the optimal value of (COP) is lower-bounded by the optimal value of this homogenized dual problem, we can sandwich the optimal values of Problems (CPP) and (COP) by those of a primal-dual pair that has been shown to exhibit strong duality. The complete proof of this result is provided in Appendix 6.1. □

In prior work, characterization of the duality gap between Problems (CPP) and (COP) has remained elusive because the feasible region of Problem (CPP) never has an interior, thus prohibiting straightforward application of Slater's constraint qualification. This result is significant because it shows that under mild conditions, the copositive formulation is exact. This means that the optimal values of Problems (MBQP) and (COP) are equivalent, so solving Problem (COP) is a valid alternative to solving Problem (MBQP). The framework developed in this paper will ultimately produce approximate solutions for Problems (CPP) and (COP), which we anticipate can be used to speed up the solution process of a purely classical solver for (MBQP). For example, the heuristic solutions or cuts used to generate them might be used to warm-start or initialize a purely classical solver.

While Problems (CPP) and (COP) are both convex, neither resolve the difficulty of Problem (MBQP) as even checking complete positivity (resp. copositivity) of a matrix is NP-hard (resp. co-NP-complete) [44]. Instead, they should be viewed as "packaging" the complexity of the problem entirely in the copositivity/complete positivity constraint. There are a number of classical approaches for (approximately) solving copositive/completely positive programs directly, such as the sum of squares hierarchy [45, 46], feasible descent method in the completely positive cone, approximations of the copositive cone by a sequence of polyhedral inner and outer approximations, among others [47, 48, 49]. In this paper, we will exploit the innate synergy between checking copositivity, which is most naturally posed as a quadratic minimization problem, and solving Ising problems. This perspective is suggestive of a hybrid quantum-classical approach where the quantum computer is responsible for checking feasibility (i.e., the "hard part") of the copositive program while the classical computer directs the search towards efficiently reducing the search space.

3.2. Cutting-Plane/Localization Algorithms. Cutting-plane/localization algorithms are convex optimization algorithms that divide labor between checking feasibility, abstracted as a *separation oracle*, and optimization of the objective [1]. In this section, we provide a high-level overview of each algorithmic step and summarize both the runtime and oracle complexities of several well-known variants; these complexity measures will ultimately correspond to the complexity of the sub-routine handled by the classical computer and the number of calls to the Ising solver, respectively.

While cutting-plane algorithms are often used to solve both constrained and unconstrained optimization problems, they are generally evaluated in terms of their complexity when solving the *feasibility problem*.

[1] The term "cutting-plane algorithm" overloaded in the literature, with one class referring very explicitly to those designed for convex/quasi-convex optimization problems (for a pedagogical reference, we refer the interested reader to [50]) and the second referring more broadly to algorithms that iteratively generate cuts (including algorithms for integer programming and non-convex optimization). In this work, we refer specifically to those designed for convex/quasi-convex optimization.

Definition 3.2 (Feasibility Problem). *For a set of interest $S \subset \mathbb{R}^m$, which can only be accessed through a separation oracle, the feasibility problem is concerned with either finding a point in the set $x \in S$ or proving that S does not contain a ball of radius r.*

Definition 3.3 (Separation Oracle). *A separation oracle for a set S, $\texttt{Oracle}_S(\cdot)$ takes as input a point $x \in \mathbb{R}^m$ and either returns $\texttt{True}$ if $x \in S$ or a separating hyperplane if $x \notin S$. A separating hyperplane is defined by a vector, $a \in \mathbb{R}^m$ and scalar $b \in \mathbb{R}$ such that $a^\top s \leq b$ for all $s \in S$ but $a^\top x \geq b$.*

The feasibility problem formulation is non-restrictive because these methods can be readily adapted to solving quasi-convex optimization problems with only a simple modification to the separation oracle. In particular, if the separation oracle indicates feasibility and returns a vector $g \in \mathbb{R}^m$ where any vectors $x, y \in \mathbb{R}^m$ with $f(y) < f(x)$ implies that $g^\top y \geq g^\top x$, this serves as a separating hyperplane for the subset of the feasible region that has a better objective than the test point. If f is subdifferentiable, any subgradient $g \in \partial f(x)$ satisfies this condition, and for Problem (COP), choosing g as the objective's coefficient vector is sufficient.

Although there are many variations of cutting-plane algorithms, at a high level, they follow a standard template that consists of alternating between checking feasibility of a test point, updating an outer approximation of the feasible region, and judiciously selecting the next test point. This standard template is summarized in Algorithm 1. An overview of the Ellipsoid algorithm is included in the Appendix as a representative example of cutting-plane algorithms, and we direct the interested reader to the references listed in Table 1 for specific implementation details. By choosing subsequent test points to be the center of the outer approximation, the algorithm is guaranteed to make consistent progress in reducing the search space (where the metric of progress may also vary across cutting-plane algorithms). Intuitively, cutting plane algorithms can be considered a high-dimensional analog of binary search. We note that the requirement $\mathrm{Vol}(S_0) \leq R$ means that the initial set must be bounded. While this is a standard assumption in the cutting-plane literature, finding such an S_0 may be non-trivial. Procedurally, one may construct a bounded outer approximation using a linear program as in Step 1, Algorithm 1 of [51].

Algorithm 1: Cutting-plane meta-algorithm (feasibility problem)

```
Input: $S_0 \subseteq \mathbb{R}^m$ (Initial Set) with $\texttt{Vol}(S_0) \leq R$
Output: $x \in S$ or False if S does not contain a ball of volume $r$
$x \leftarrow \texttt{Center}(S_0)$;
$k \leftarrow 0$;
while Oracle(x) is not True and $\texttt{Vol}(S_k) \geq r$ do
    $S_{k+1} \leftarrow \texttt{Add\_Cut}(S_k, \texttt{Oracle}(x))$;
    $x \leftarrow \texttt{Center}(S_{k+1})$;
    $k \leftarrow k + 1$;
end
if Oracle(x) is True then
    return $x$;
else
    return False;
end
```

A number of well-known variants of cutting-plane algorithms are summarized in Table 1. Differences across instantiations of cutting-plane algorithms vary in how subsequent test points are chosen, how the outer approximation is updated, and how progress in decreasing the outer approximation's size is measured. Each of the surveyed variants strikes a different balance between the computational effort needed to compute a good center versus the resolution used to represent the outer approximation. Critically, except for the Center of Gravity method, all cutting-plane algorithms summarized in Table 1 have a polynomial complexity in the dimension of the optimization variables in terms of both oracle queries and total runtime excluding the oracle calls (i.e., the total complexity of adding the cuts and generating test points). This suggests that if a cutting-plane algorithm were applied to Problem (COP), the complexity of the problem is offloaded onto the separation oracle; this is the subroutine we propose to handle using an Ising solver.

Name	Oracle Queries	Total runtime (excluding oracle queries)	References
Center of Gravity	$\mathcal{O}(m\log(\frac{R}{r}))$	#P-hard [52]	[53]
Ellipsoid	$\mathcal{O}(m^2\log(m\frac{R}{r}))$	$\mathcal{O}(m^4\log(m\frac{R}{r}))$	[54, 55, 56]
Inscribed Ellipsoid	$\mathcal{O}(m\log(m\frac{R}{r}))$	$\mathcal{O}((m\log(m\frac{R}{r}))^{4.5})$	[57, 58]
Volumetric Center	$\mathcal{O}(m\log(m\frac{R}{r}))$	$\mathcal{O}(m^{1+\omega}\log(m\frac{R}{r}))$	[59]
Analytic Center	$\mathcal{O}(m\log^2(m\frac{R}{r}))$	$\mathcal{O}(m^{1+\omega}\log^2(m\frac{R}{r}) + (m\log(m\frac{R}{r}))^{2+\frac{\omega}{2}})$	[60]
Random Walk	$\mathcal{O}(m\log(m\frac{R}{r}))$	$\mathcal{O}(m^7\log(m\frac{R}{r}))$	[61]
Lee, Sidford, Wong	$\mathcal{O}(m\log(m\frac{R}{r}))$	$\mathcal{O}(m^3\log^{\mathcal{O}(1)}(m\frac{R}{r}))$	[62]

Table 1: This table summarizes the number of oracle queries and total runtime guarantees of a number of well-known cutting-plane variants. The stated runtimes are in terms of the problem dimension, m, the volume of the initial set, R, and the minimum volume of the set of interest, r. The constant ω represents the fast matrix multiplication constant.

3.3. Application to Copositive Optimization. Now that we have introduced cutting-plane algorithms, we are in a position to discuss their application to the copositive program (COP). First, we will show how a *copositivity oracle* can be readily transformed into a separation oracle for the feasible region of Problem (COP). We will conclude with a discussion of how a copositivity oracle can be implemented using an Ising solver. Formally, we define a copositivity oracle as follows:

Definition 3.4 (Copositivity Oracle). *A copositivity oracle takes as input a matrix, M, and either returns* `True` *if M is copositive or returns a vector $z \in \mathbb{R}^n_{\geq 0}$ such that $z^\top M z < 0$ (a "certificate of non-copositivity").*

A copositivity oracle can be turned into a separation oracle for the feasible region of Problem (COP) by expanding the terms in $z^\top M(\hat{\mu}, \hat{\lambda}, \hat{\gamma})z$. Explicitly, a test point, $(\hat{\mu}, \hat{\lambda}, \hat{\gamma})$, is infeasible if and only if $M(\hat{\mu}, \hat{\lambda}, \hat{\gamma})$ is not copositive. Given $M(\hat{\mu}, \hat{\lambda}, \hat{\gamma})$ as input, the copositivity oracle returns a certificate of non-copositivity $z \in \mathbb{R}^{n+1}_{\geq 0}$ such that $z^\top M(\hat{\mu}, \hat{\lambda}, \hat{\gamma})z < 0$. In contrast, feasibility means that $z^\top M(\mu, \lambda, \gamma)z \geq 0$. Equivalently, the halfspace defined by

$$b = z^\top \begin{pmatrix} Q & c \\ c^\top & \cdot \end{pmatrix} z, \tag{3.2}$$

$$a[\mu_i^{(\text{lin})}] = z^\top \begin{pmatrix} \cdot & \frac{1}{2}A_{i,*}^\top \\ \frac{1}{2}A_{i,*} & \cdot \end{pmatrix} z, \tag{3.3}$$

$$a[\mu_i^{(\text{quad})}] = z^\top \begin{pmatrix} A_{i,*}^\top A_{i,*} & \cdot \\ \cdot & \cdot \end{pmatrix} z, \tag{3.4}$$

$$a[\lambda_j] = z^\top \begin{pmatrix} -\mathbb{1}_{\{j\}}\mathbb{1}_{\{j\}}^\top & \frac{1}{2}\mathbb{1}_{\{j\}} \\ \frac{1}{2}\mathbb{1}_{\{j\}}^\top & \cdot \end{pmatrix} z, \tag{3.5}$$

$$a[\gamma] = z^\top \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix} z, \tag{3.6}$$

is a separating hyperplane for $(\hat{\mu}, \hat{\lambda}, \hat{\gamma})$, where we use symbolic indexing to explicitly denote which variable each coefficient corresponds to. Explicitly, the inner product between a and (μ, λ, γ) is given by

$$a^\top(\mu, \lambda, \gamma) = \sum_i a[\mu_i^{(\text{lin})}]\mu_i^{(\text{lin})} + \sum_i a[\mu_i^{(\text{quad})}]\mu_i^{(\text{quad})} + \sum_j a[\lambda_j]\lambda_j + a[\gamma]\gamma. \tag{3.7}$$

This shows that given a copositivity oracle, constructing a separation oracle for Problem (COP), of dimension $\mathcal{O}(m)$ and copositivity constraints on matrices of size $\mathcal{O}(n)$, entails evaluating $\mathcal{O}(m)$ vector-matrix-vector products, each of dimension $\mathcal{O}(n)$. The cutting-plane algorithms presented in Section 3.2 can then be applied without further modification.

We note that the application of cutting-plane algorithms to copositive optimization has been explored from a classical perspective in [63], which considered their application to discrete markets and games, and [64], which applied the algorithm to detect complete positivity of matrices. We believe our work is complementary to these prior works. While our work relies on the off-the-shelf application of well-known cutting-plane variants, the algorithmic modifications in [63], and [64] provide insight for further improving our framework. For example, the cutting plane algorithm in [63] is readily hybridized leveraging our proposed approach. We do not explore this extension in this paper due to their lack of convergence guarantees, while those of the variants presented are central to our theoretical analysis. Moreover, the problem settings considered in these works serve as inspiration for additional applications that can be addressed with Ising solvers. On the other hand, the proof of strong duality (Theorem 3.1) can be applied to address questions that were left open in [63]. We emphasize that the contribution of this work is not the copositive reformulation or a novel cutting-plane algorithm but rather the insight that these ideas are synergistic with recent advances in quantum(-inspired) computing. In particular, copositive optimization is useful for deriving and analyzing new hybrid algorithms rooted in existing convex optimization algorithms, thus filling a gap in the hybrid algorithms literature.

3.4. QUBO Approximation of Copositivity Checks. Checking copositivity of $M(\mu, \lambda, \gamma)$ is naturally posed as the following (possibly non-convex) quadratic minimization problem

$$\begin{aligned} \underset{z \in \mathbb{R}^{n+1}_{\geq 0}}{\text{minimize}} \quad & z^\top M(\mu, \lambda, \gamma) z \\ \text{subject to} \quad & ||z||_p \leq 1, \end{aligned} \tag{3.8}$$

where a matrix is copositive if and only if min (3.8) is non-negative[2]. There are several alternative approaches for checking copositivity [65, 66, 67, 68, 69]; however, they are typically derived with Problem (3.8) as the starting point and designed to exploit particular properties of Problem (3.8). By choosing $p = \infty$, Problem (3.8) can be approximated by a QUBO where an approximation of the matrix M, $\hat{M}$, is used such that the optimization variables $\hat{z}$ represent a binary expansion of z with k bits as follows:

$$\begin{aligned} \underset{\hat{z}}{\text{minimize}} \quad & \hat{z}^\top \hat{M}(\mu, \lambda, \gamma) \hat{z} \\ \text{subject to} \quad & \hat{z} \in \{0, 1\}^{k(n+1)}. \end{aligned} \tag{QUBO}$$

Explicitly, $\hat{M}(\mu, \lambda, \gamma)$ and $M(\mu, \lambda, \gamma)$ are related as follows:

$$\hat{M}(\mu, \lambda, \gamma) = \mathcal{D}^\top M(\mu, \lambda, \gamma) \mathcal{D}, \tag{3.9}$$

where

$$\mathcal{D} := \frac{1}{2^k - 1} \begin{pmatrix} 2^0 & \cdots & 2^{k-1} & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 2^0 & \cdots & 2^{k-1} & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 2^0 & \cdots & 2^{k-1} \end{pmatrix}, \tag{3.10}$$

The construction of (QUBO) is detailed in Appendix 6.2. The explicit implementation of `Oracle`$(\cdot)$ is summarized in Algorithm 2. Critically, the constraints of (3.8) are implied by the natural domain of the Ising solver, mitigating the need to tune coefficients in a penalty method carefully.

3.5. Discussion. In summary, we propose to solve Problem (MBQP) by constructing the equivalent copositive formulation in (COP) and applying any variant of Algorithm 1. Within Algorithm 1, the implementation of `Oracle`$(\cdot)$ is specified by Algorithm 2. This process is depicted in Figure 1. Now that we have presented our method in full, several comments are in order.

Computational complexity. While the stated complexity of the cutting-plane algorithms is applicable to any problem, it is suggestively stated in terms of the variable m. This notational overload is a deliberate choice because the dimension of the dual copositive program is equal to the total number of constraints in Problem (CPP), which is $2m + |B| + 1 = \mathcal{O}(m)$. The number of constraints can be reduced to $m + |B| + 1$ using the homogenized completely positive reformulation presented in Appendix 6.1. While this will have no impact on the asymptotic complexity of the method, it can result in a practical reduction in runtime. If $\mathcal{T}_Q$ represents the oracle complexity of a particular method, the total additional overhead of converting the copositivity oracle into a separation oracle is given by $\mathcal{O}(mn^2\mathcal{T}_Q)$.

[2] While copositivity is defined as a condition over all of $\mathbb{R}^{n+1}_{\geq 0}$, quadratic scaling of the objective ensures that optimizing over a norm ball is sufficient for detecting copositivity.

Algorithm 2: Separation oracle, `Oracle`$(\cdot)$

```
Input: $(\hat{\mu}, \hat{\lambda}, \hat{\gamma})$ (Test point)
Output:
    $\begin{cases} \texttt{True} & \text{if } (\hat{\mu}, \hat{\lambda}, \hat{\gamma}) \text{ is feasible} \\ \text{Separating hyperplane for } (\hat{\mu}, \hat{\lambda}, \hat{\gamma}) & \text{otherwise} \end{cases}$

// Solve (QUBO) using an Ising solver

    $z^* \leftarrow \arg\min_{\hat{z}} \quad \text{(QUBO)}$

if $\min\text{(QUBO)} \geq 0$ then
    return True;
else
    (3.11)    $z = \mathcal{D} z^*$
    (3.12)    $b = z^\top \begin{pmatrix} Q & c \\ c^\top & \cdot \end{pmatrix} z$
    (3.13)    $a[\mu_i^{(\text{lin})}] = z^\top \begin{pmatrix} \cdot & \frac{1}{2} A_{i,*}^\top \\ \frac{1}{2} A_{i,*} & \cdot \end{pmatrix} z$
    (3.14)    $a[\mu_i^{(\text{quad})}] = z^\top \begin{pmatrix} A_{i,*}^\top A_{i,*} & \cdot \\ \cdot & \cdot \end{pmatrix} z$
    (3.15)    $a[\lambda_j] = z^\top \begin{pmatrix} -\mathbb{1}_{\{j\}} \mathbb{1}_{\{j\}}^\top & \frac{1}{2} \mathbb{1}_{\{j\}} \\ \frac{1}{2} \mathbb{1}_{\{j\}}^\top & \cdot \end{pmatrix} z$
    (3.16)    $a[\gamma] = z^\top \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix} z$

    return $a$, $b$;
end
```

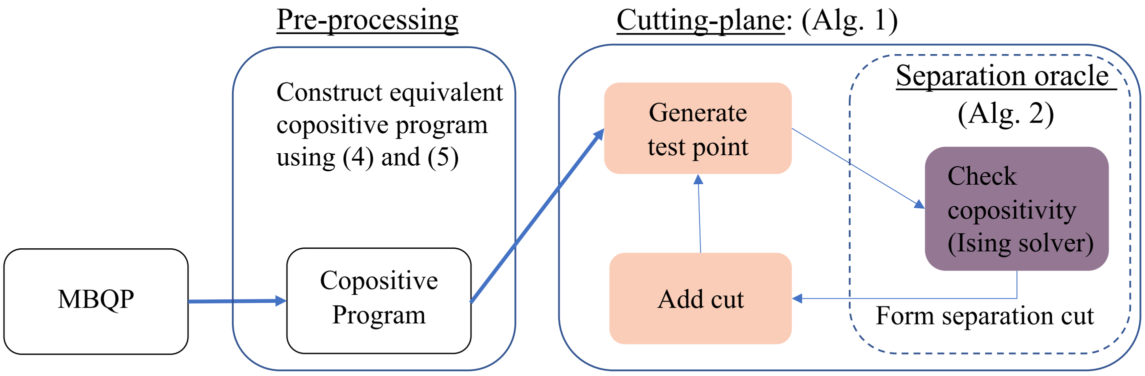

Pre-processing
Cutting-plane: (Alg. 1)
Construct equivalent copositive program using (4) and (5)
MBQP
Copositive Program
Generate test point
Add cut
Separation oracle (Alg. 2)
Check copositivity (Ising solver)
Form separation cut

Fig. 1: This figure depicts the entire solution process for solving a MBQP of the form (MBQP).

Discretization size. Discretization of the copositivity check automatically introduces an approximation to the copositivity checks. The approximation fidelity is improved as the number of discretization points is increased, although it is limited by the hardware. Not only does representing a finer discretization require more qubits, but it also results in a greater skew in the coefficients of the Ising Hamiltonian. This becomes challenging since many existing hardware platforms have limited precision in their implementable couplings. In contrast, too coarse of a discretization runs the risk of missing the certificate of non-copositivity entirely. This suggests that the discretization scheme should be well-tailored to the problem at hand; Appendix 6.2 provides guidance for choosing a discretization size based on the coefficients of the Ising Hamiltonian. A promising alternative is to circumvent discretization entirely, and apply quantum(-inspired) solvers that natively solve continuous variable box-constrained quadratic programs, such as the coherent continuous-variable machine (CCVM) recently proposed in [70].

Multiple cuts. Following standard convention, this work assumes that the copositivity oracle returns a single value. In contrast, in practice, many of the aforementioned Ising solvers are heuristics that involve multiple readouts. Each of these reads can be used to construct a cut, where negative, zero, and positive Ising objective values correspond to deep, neutral, and shallow cuts, respectively. Adding multiple cuts during each iteration is a possible heuristic for improving the convergence rate of the cutting-plane algorithm. While the true ground state corresponds to the deepest cut, the convergence rate guarantees stated in Table 1 hold so long as a neutral or deep cut is added at each iteration. Consequently, the proposed approach is not overly reliant on the Ising solver's ability to identify the ground state and is resilient to heuristics. Critically, this raises the question of how to proceed if the Ising solver fails to return a certificate of non-copositivity, which will likely depend on problem specifics, such as the current outer approximation, the objective values of the samples, and the solver itself. For example, if the Ising solver returns positive but small solutions, depending on the current outer approximation, the addition of shallow cuts can still reduce the search space. On the other hand, if all non-zero solutions result in a large objective value, one could increase confidence that the test point is feasible by increasing the number of discretization points and readouts.

4. Experiments. We conducted an investigation of the proposed method on the maximum clique problem, which finds the largest complete subgraph of a graph. Given a graph, the maximum clique problem can be formulated as a completely positive program

$$\begin{aligned} \underset{X \in \mathbb{R}^{n\times n}}{\text{maximize}} \quad & \left\langle \mathbb{1}\mathbb{1}^\top,\, X \right\rangle \\ \text{subject to} \quad & \left\langle \overline{A} + I,\, X \right\rangle = 1, \\ & X \in \mathcal{C}_n^*, \end{aligned} \tag{4.1}$$

where $\overline{A}$ is the adjacency matrix of the graph's complement [71]. We note that solving the maximum clique problem is equivalent to solving the maximum independent set problem on the complement graph. The dual of (4.1) is the following copositive program:

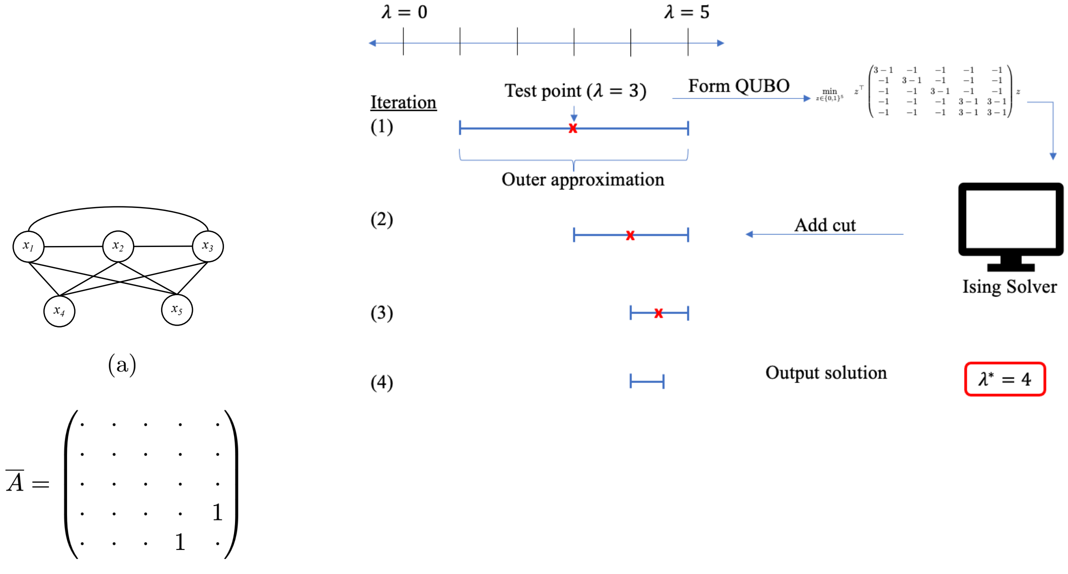
$$\begin{aligned} \underset{\lambda \in \mathbb{R}}{\text{minimize}} \quad & \lambda \\ \text{subject to} \quad & \lambda(I + \overline{A}) - \mathbb{1}\mathbb{1}^\top \in \mathcal{C}_n. \end{aligned} \tag{4.2}$$

This copositive program only has one variable regardless of the graph's number of vertices or edges. Thus, we solve it with bisection, a special case of the ellipsoid algorithm, as the cutting-plane algorithm. The copositivity check's size is determined by the number of vertices, n, which impacts the complexity of computing the cuts from the certificates of non-copositivity. The number of edges can be used to upper-bound the size of the maximum clique, thus determining the size of the initial feasible region; however, its effect on the complexity of checking copositivity is unclear.

4.1. QUBO Subroutine. Many hybrid algorithms are designed by replacing subroutines of existing (fully classical) algorithms with a quantum(-inspired) counterpart. An oft-neglected consideration is whether the quantum(-inspired) computer is applied to a bottleneck in the original algorithm. We contend that for a hybrid algorithm to yield significant speed-ups, the quantized subroutine should constitute the bulk of the algorithm's complexity. In keeping with this supposition, we first evaluated whether the copositive cutting-plane algorithm shifts the complexity of the solution process onto the copositivity checks by profiling each component of the algorithm separately.

To study the scaling of each component of the proposed approach, we considered random max-clique problems with up to $10, 30, \ldots, 270$ vertices, where the maximum graph size varies by edge density to ensure a reasonable computation time for this experiment. For each graph size, we generated 25 random Erdős-Renyi instances with edge densities $p \in \{0.25,\, 0.5,\, 0.75\}$ and solved to an absolute gap of 0.9999 between upper and lower bounds (because the optimal solution is known to be integral) using the proposed copositive cutting plane algorithm. The copositivity checks were conducted by solving Anstreicher's MILP characterization of copositivity [65] (which we found to be one of the most competitive classical formulations), using `Gurobi` version 9.0.3 [72]. [3]

[3] All experiments were run on an AMD Ryzen 7 1800X Eight-Core Processor@3.6GHz with 64GB of RAM and 16 threads. All code needed to reproduce these experiments is available at https://github.com/StanfordASL/copositive-cutting-plane-max-clique.



(a)

(b)

(c)

Fig. 2: Figure 2a depicts a small maximum clique example where there are edges between all vertices except x_4 and x_5. Figure 2b depicts the adjacency matrix of graph 2a's complement, which has a single edge between vertices x_4 and x_5. Figure 2c depicts the solution process for the copositive cutting-plane algorithm.

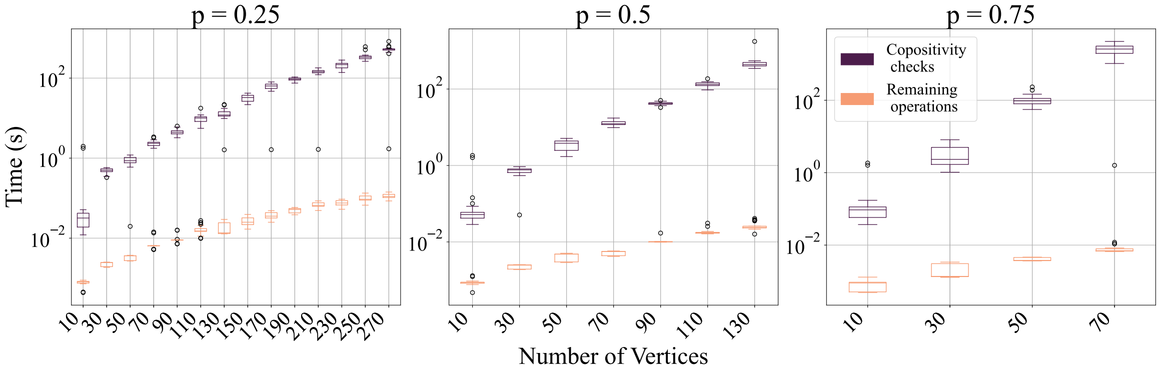


Fig. 3: This figure plots the time spent on the copositivity checks versus all other operations in the proposed method. The copositivity checks grow exponentially with the number of vertices, while the other operations grow modestly.

Figure 3 plots the time the copositive cutting plane algorithm spent on the copositivity checks versus other operations (updating the outer approximation and computing test points). The time spent on the copositivity checks scales exponentially with the number of vertices in the graph, while the time spent on other operations grows modestly. This is because Problem (4.1) only has one constraint regardless of the graph's number of vertices or edges. In contrast, the size of the copositivity check is exactly equal to the number of vertices in the graph. Both the theoretical analysis and empirical results confirm that the proposed approach shifts the complexity of the copositive program onto the copositivity checks. This experiment shows that the proposed methodology is particularly effective for problems whose constraints remain constant or grow modestly with problem size.

While undesirable for a fully classical implementation, the overwhelming complexity in the copos-

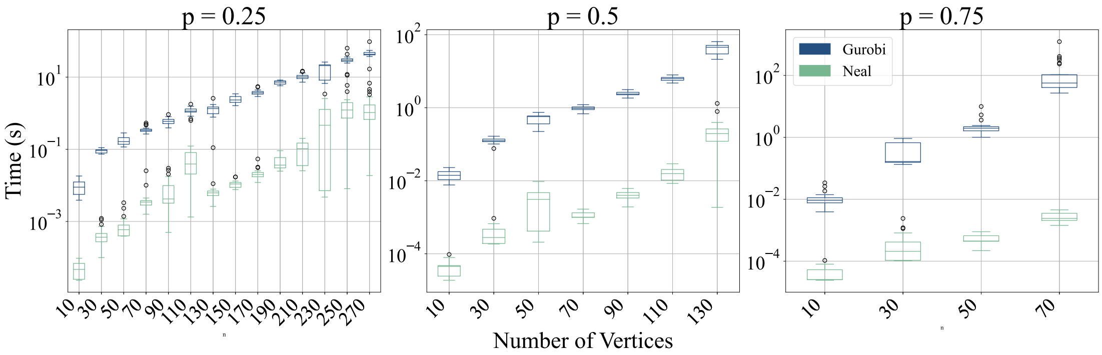


Fig. 4: This figure plots the time to target to 99% confidence of the Simulated Annealing (SA) implementation in `Neal` (replacing the continuous feasible region, $[0,1]^n$, with its vertices, $\{0,1\}^n$) against the solution time of `Gurobi` for the copositivity checks. We solved each copositivity check with 100 sweeps and 1000 reads. For all densities, both methods scale exponentially with the number of vertices in the graph; however, SA is several orders of magnitude faster than `Gurobi`.

itivity checks represents an opportunity for a quantum(-inspired) solver to beget significant speed-up. They will result from being able to execute the copositivity checks faster than the classical implementation (i.e., Anstreicher's MILP formulation). To investigate potential speedups from using a stochastic Ising solver, we re-solved each copositivity check that yielded a certificate of non-copositivity using Simulated Annealing (SA) through the software `Neal` version 0.5.9, a SA sampler [73], i.e., a solver that returns samples on the solutions distribution generated by SA. Because SA is not guaranteed to find the global optima in a single annealing cycle, we define a probabilistic notion of time to target. In particular, we follow [74] and define the time to target with s confidence to be the number of repetitions to find the ground state at least once with probability s multiplied by the time for each annealing cycle, $\mathtt{T}_{\text{anneal}}$, i.e.,

$$\mathtt{TTT}_s = \mathtt{T}_{\text{anneal}} \frac{\log(1-s)}{\log(1-\hat{p}_{\text{succ}})}, \tag{4.3}$$

where $\hat{p}_{\text{succ}}$ is the expected value of the returned solution divided by the ground state/minimum. This results in a probability of success that interpolates between counting only solutions corresponding to the ground state and counting all certificates of non-copositivity as successes by considering the relative quality of each sample. We will also consider analogous scenarios where only ground state solutions are counted as success; we reserve the terminology "time to solution", $\mathtt{TTS}_s = \mathtt{T}_{\text{anneal}} \frac{\log(1-s)}{\log(1-p_{\text{succ}})}$, for such cases to distinguish from the previously defined time to target. The values of $\hat{p}_{\text{succ}}$ and p_{succ} is evaluated empirically over 1000 samples/reads. The time per annealing cycle, $\mathtt{T}_{\text{anneal}}$, was evaluated as the total wall-clock time (for all reads) divided by the number of reads. All other `Neal` parameters were left as their default values.

The time to solution and time to target metrics are intended to facilitate comparison between deterministic and stochastic solvers by taking into account both the time needed to run the stochastic solver and its probability of success. It is impossible to guarantee 100% success for a stochastic solver under this formulation (mathematically, this would be equivalent to trying to compute $\mathtt{TTS}_1$ or $\mathtt{TTT}_1$, which are both undefined). However, computing these metrics with a high degree of certainty (e.g., $s = 1-\epsilon$) is widely accepted as a tolerable, albeit imperfect, benchmark [75]. In the remainder of this section, we will compare time to target metrics from SA against solution time from `Gurobi`, but the astute reader should keep in mind that the two solvers serve fundamentally different purposes. This comparison between the solvers is not intended to be interpreted in isolation, but rather to highlight where it might be appropriate and beneficial to substitute a heuristic Ising solver within the copositive cutting-plane framework.

For each copositivity check solved, we considered discretizations corresponding to $\min_{\hat{z}\in\{0,1\}^n} \hat{z}^\top M \hat{z}$ (i.e., no additional problem discretization). We solved each copositivity check with 100 sweeps and 1000 reads[4]. Figure 4 plots the time to target with 99% confidence from `Neal`

[4] While the performance of `Neal` depends on the number of sweeps, we found that optimizing the number of sweeps does not result in significant reductions in the time to target. We provide further discussion in the Appendix 6.3.

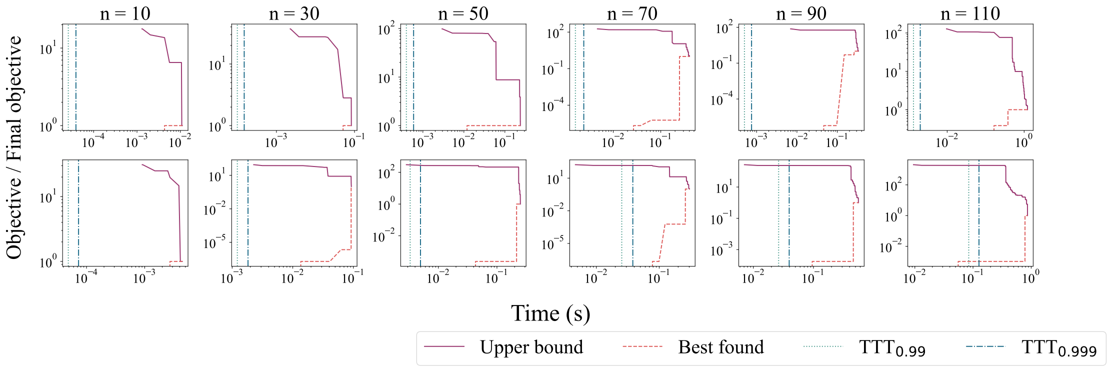


Fig. 5: This figure depicts sample trajectories of `Gurobi`'s upper and lower bounds against $\texttt{TTT}_{0.99}$ and $\texttt{TTT}_{0.999}$ for edge density $p = 0.25$. For each graph size, the top row represents the instance where the ratio between `Gurobi`'s solution time and $\texttt{TTT}_{0.99}$ is the greatest, and the bottom row represents the instance where the ratio is the smallest–all instances were run with 100 sweeps. In most instances, `Neal` reaches the $\texttt{TTT}_{0.999}$ confidence before `Gurobi` even returns a callback.

against the solution time from `Gurobi`[5]. We see that for all graph sizes, `Neal` can consistently find certificates of non-copositivity in orders of magnitude less time than `Gurobi`. Notably, `Neal` and `Gurobi` demonstrate similar scaling with respect to the number of vertices.

Unlike SA, which operates without reference to rigorous optimality bounds, `Gurobi`'s solution process tracks both upper and lower bounds on the objective value and terminates only when they reach user-specified stopping conditions. To evaluate whether the optimal objective is found early in the solution process and time is spent closing the upper bounds, we plotted `Gurobi`'s lower and upper bounds progress against time together with $\texttt{TTT}_{0.99}$ and $\texttt{TTT}_{0.999}$ in Figure 5 for instances with density $p = 0.25$. Analogous plots for other densities are included in the Appendix. For each graph size, we plotted the instances where the ratio between `Gurobi`'s solution time and $\texttt{TTT}_{0.99}$ is the greatest (top row) and least (bottom row)–all instances were run with 100 sweeps. For each instance, we plot `Gurobi`'s upper bound (purple, solid) and best objective found (red, dashed), and `Neal` $\texttt{TTT}_{0.99}$ (light teal, dashed), and $\texttt{TTT}_{0.999}$ (dark teal, dash-dot) (colored figures are available online). We found that in most instances, `Neal` reaches the time to target with 99.9% confidence before `Gurobi` even returns a callback (i.e., when the dark teal, dash-dot line does not intersect either of the purple-solid or red-dashed lines); this is likely due to an initial pre-processing step. Pre-processing is a necessary overhead for Gurobi's intended purpose of proving optimality, thus hampering its relative performance on smaller or easier problems. Critically, in the proposed approach, optimality guarantees are only necessary for cases where the test point is copositive, while the algorithm can make progress with any certificate of non-copositivity, even if it is not globally optimal. This suggests that heuristics (e.g., Ising solvers) and complete methods (e.g., `Gurobi`) could play complementary roles within the same copositive cutting-plane algorithm. For example, for some test points, one may avoid the overhead of `Gurobi` altogether if the Ising solver quickly returns a certificate of non-copositivity. On the other hand, if the Ising solver fails to generate such a certificate, one may rely on `Gurobi` for proving that the test point is copositive.

4.2. Overall Cutting-Plane Algorithm. Next, we compared the copositive cutting-plane algorithm with the SA implementation in `Neal` as the Ising solver against solving a mixed-integer linear program (MILP) formulation of maximum-clique directly with `Gurobi`. `Gurobi`'s solution time was evaluated on the following MILP formulation of maximum clique:

$$\begin{aligned} \underset{x \in \{0,1\}^n}{\text{maximize}} \quad & \mathbb{1}^\top x \\ \text{subject to} \quad & x_i + x_j \leq 1, \quad \forall (i,j) \in \overline{E}, \end{aligned} \tag{4.4}$$

[5] Note that `Gurobi`'s solution time in Figure 4 is different from copositivity checks profiling in Figure 3. This is because only non-copositive instances were considered for this comparison, while all instances, including copositive ones, were included in the profiling comparison.

where $\overline{E}$ is the edges in the complement graph. This is a MILP with n binary variables, where n is the number of vertices in the graph, and $|\overline{E}|$ constraints (i.e., the number of edges in the complement graph). The copositive cutting-plane algorithm was tested with different sweeps and reads, which were fixed throughout each run of the algorithm. The solid pink lines in Figure 6 plot the runtime of the copositive cutting-plane algorithm for a representative set of these parameters. Because `Neal` may fail to find a certificate for some non-copositive matrices, this method may incorrectly reduce the upper bound in the outer approximation; however, it cannot incorrectly update the lower bound. Throughout the algorithm, we track the exact lower bound and the approximate upper bound (which is updated when `Neal` fails to find a certificate of non-copositivity). The algorithm is terminated when the lower bound and approximate upper bound are within an absolute tolerance of 0.9999; this is the same stopping condition as the exact case where we checked copositivity using Anstreicher's MILP formulation. Consequently, the solution returned was determined by rounding the lower bound up to the nearest integer. This means that the solution returned is guaranteed to be a lower bound for the maximum clique instance. The fraction of correct maximum clique solutions is indicated by the color of the markers.

From the pink plots, we see that for a *fixed parameter setting*, the copositive cutting-plane algorithm exhibits polynomial scaling in the graph size. While it is tempting to extrapolate this scaling relationship to larger graph sizes, the failure of parameters that were successful on smaller graphs on larger graph sizes (denoted by the green and blue markers) indicate that it is unlikely that fixed parameter settings will continue to be effective *ad infinitum*. In particular, we found that while smaller and sparser instances are tolerant of fewer sweeps and reads (resulting in shorter calls to the Ising solver), larger instances required more sweeps and reads to be accurate. While the copositive cutting-plane algorithm is designed to benefit from speed-ups of state-of-the-art Ising solvers, the converse is also expected; poor scaling of the Ising solver will make its way into the runtime as the time spent in each oracle call, $\mathcal{T}_Q$.

On the other hand, we observe that the confidence intervals for `Neal` are significantly wider than those of `Gurobi` and the copositive cutting-plane algorithm. This is potentially due to the sensitivity of `Neal` to its penalty weight. While `Neal` can also produce lower bounds for the maximum clique, the bounds can only be updated when it returns solution corresponds to a clique. In contrast, the copositive cutting-plane algorithm can generate cuts from any certificate of non-copositivity, even those that do not correspond to a clique at all. This means that the copositive cutting-plane algorithm can make progress from a larger set of the solutions returned by the Ising solver. This suggests that the copositive cutting-plane algorithm may even be competitive against its underlying copositivity checker solving a direct formulation of the problem, particularly if its performance is highly sensitive to parameter settings.

We note that `Gurobi` takes advantage of multi-threading while neither `Neal` nor the copositive cutting-plane algorithm do. This raises the important question of how much the copositive cutting-plane algorithm (and `Neal`) will benefit from similar decomposition and parallelization efforts.

Finally, we investigated the effectiveness of directly converting the maximum clique problem to an Ising problem using a standard penalty formulation. To do so, we solved each of the maximum clique problem instances using the `maximum_clique` formulator[6] with `Neal` as the sampler and a range of penalty weights in $\{2^{-1}, 2^{0}, \ldots, 2^{4}\}$; the number of sweeps was left to its default value of 1000. This results in a QUBO with n variables and $|\overline{E}|$ quadratic terms. For each instance, we conducted 1000 reads and evaluated the average normalized sample size (the size of the returned solution divided by the ground truth maximum clique size) and the fraction of reads that resulted in a valid clique; a ground state solution is one that is both a valid clique and has a normalized sample size of 1. We computed the probability of success, p_{succ}, as the fraction of reads that resulted in a ground state solution, which was subsequently used to derive the time to solution to 99.9% confidence. Figure 7 plots each of these metrics as a function of the penalty weights and graph size for edge density $p = 0.25$. Analogous plots for other densities are included in the Appendix.

For penalty weights 0.5 and 1, the normalized sample size is often greater than 1, resulting in samples that do not represent a valid clique. For penalty weights $2, 4, 8$, and 16, most samples were valid cliques; however, the normalized sample sizes were typically less than 1; these represent non-maximum cliques. Generally, as the penalty weight is increased, the normalized sample size decreases, and the fraction of valid cliques increases. This aligns with the interpretation that the penalty weight represents a trade-off between satisfying the constraints versus optimizing the objective. These

[6] https://docs.ocean.dwavesys.com/projects/dwave-networkx/en/latest/reference/algorithms/generated/dwave_networkx.maximum_clique.html

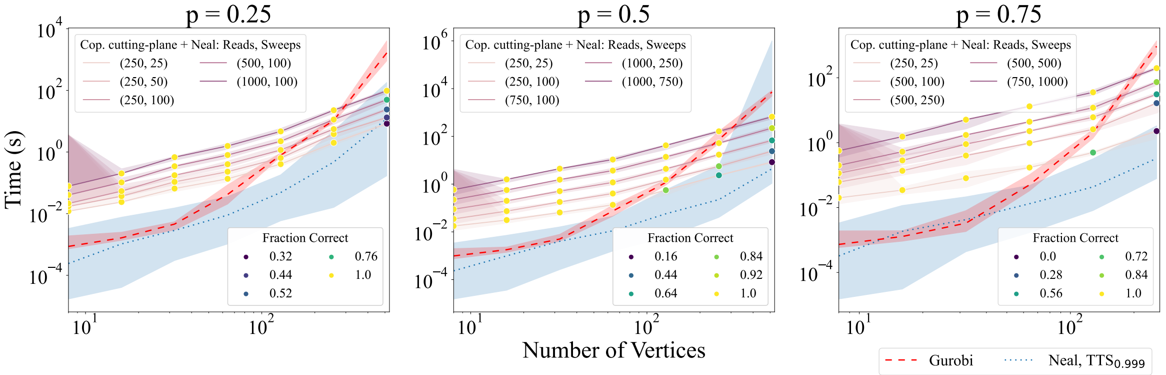


Fig. 6: This figure plots the solution time for the copositive cutting-plane algorithm with the Simulated Annealing implementation in `Neal` as the Ising solver (with various fixed parameters), the solution time when solving a mixed-integer programming (MILP) formulation of maximum-clique directly with `Gurobi`, and the corresponding $\mathrm{TTT}_{0.999}$ from `Neal` applied to the `maximum_clique` formulation with penalty weight 1. While `Gurobi`'s solution time is orders of magnitude faster than the copositive cutting-plane algorithm for the smallest graph sizes, the copositive cutting-plane algorithm starts outperforming it for larger graph sizes.

empirical results also corroborate the analytical results of [76], which state that the minimum valid penalty weight for the stable set of a graph is 1. Given that `maximum_clique` represents the maximum clique problem as finding the stable set of the graph built with the complement of the original edges, the bound on the penalty weight is valid. This experiment demonstrates that while the penalty formulation may be an effective heuristic, it typically requires carefully tuning the penalty weights to optimize the trade-off between satisfying the constraints and optimizing the objective. Figure 6 also plots the corresponding $\mathrm{TTT}_{0.999}$ from `Neal` applied directly to the `maximum_clique` formulation with penalty weight 1 against the `Gurobi` solution time and the copositive cutting-plane solution time.

5. Conclusions. In this paper, we advocate for the development of a theory of hybrid quantum-classical algorithms that analytically quantifies their performance. Metrics for comparing different hybrid algorithms may include the number of calls to the quantum computer, the complexity of the classical portion, and the requirements on the quantum computer. As a step in this direction, we demonstrate a class of hybrid algorithms for mixed-binary quadratic programming problems using Ising solvers and report the aforementioned metrics. Our framework relies on Burer's convex reformulation of such problems using completely positive programming. Our first contribution is to extend this result and show that under mild conditions, the dual copositive program exhibits strong duality. We then propose a hybrid quantum-classic solution algorithm based on cutting-plane algorithms, where an Ising solver is used to construct the separation oracle. This approach partially mitigates the heuristic nature of many state-of-the-art Ising solvers. Moreover, the runtime of the components handled by the classical computer scales polynomially with the number of constraints in the original mixed-binary quadratic program. This suggests that if our approach is applied to a problem with exponential scaling, the complexity is shifted on the subroutine carried out by the hardware accelerator, e.g., the quantum computer. Our proposed approach is particularly appealing because it suggests that the proposed approach could take advantage of any speedup that exists even without an explicit characterization of what that speedup is.

While the proposed framework seems like a promising way forward for utilizing quantum/quantum-inspired Ising solvers, a crucial question remains regarding how the algorithm should proceed if the Ising solver fails to find a certificate of non-copositivity. Could one design an efficient algorithm that circumvents the ambiguity due to failure to find a certificate of non-copositivity (perhaps using a sum-of-squares-based inner approximations of the copositive cone)? Alternatively, is there a complexity barrier that prevents such a construction? More broadly, identifying fundamental limitations, such as this example, is important for understanding what requirements should be placed on new computing architectures. While the theory in this paper is framed in the context of understanding algorithms that interact with existing hardware, we believe its most potent impact is in informing the co-design of "hardware primitives" and optimization algorithms. As a concrete example, the

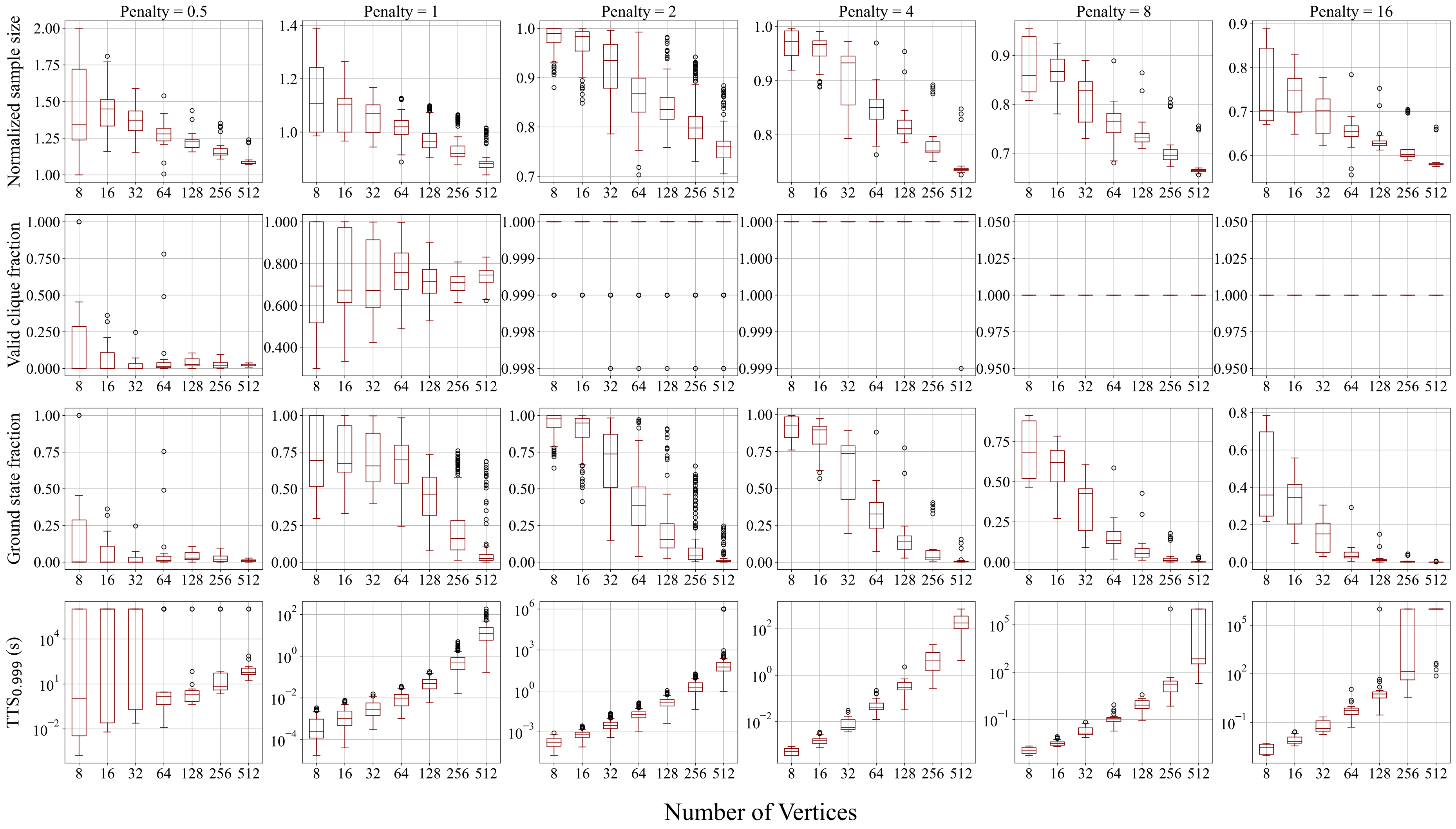


Fig. 7: This figure plots the normalized sample size (the size of the returned solution divided by the ground truth maximum clique size) and the fraction of reads that resulted in a valid clique for graph density $p = 0.25$. These figures were used to compute the fraction of reads resulting in a ground state solution and the corresponding $\mathtt{TTT}_{0.999}$ (also plotted). As the penalty weight is increased, the normalized sample size decreases, and the fraction of valid cliques increases. This highlights the delicate trade-off between constraints and the objective in penalty formulations.

coherent continuous variable machine (CCVM) recently proposed in [70] circumvents discretization of the copositivity checks entirely, thus reducing the hardware resources and potentially improving problem conditioning for each copositivity check. Not only does this hardware innovation directly benefit the optimization community, but the analysis in this work helps to justify the development of continuous variable devices.

Acknowledgements. This work was supported by NSF CCF (grant #1918549), NASA Academic Mission Services (contract NNA16BD14C – funded under SAA2-403506). R.B. acknowledges support from the NASA/USRA Feynman Quantum Academy Internship program. The authors wish to thank Aaron Lott, Luis Zuluaga, and Juan Vera for helpful input and discussions during the development of these ideas.

REFERENCES

[1] M. W. Johnson, M. H. Amin, S. Gildert, T. Lanting, F. Hamze, N. Dickson, R. Harris, A. J. Berkley, J. Johansson, P. Bunyk *et al.*, “Quantum annealing with manufactured spins,” *Nature*, vol. 473, no. 7346, pp. 194–198, 2011.

[2] E. Farhi, J. Goldstone, and S. Gutmann, “A quantum approximate optimization algorithm,” *arXiv preprint arXiv:1411.4028*, 2014.

[3] T. Honjo, T. Sonobe, K. Inaba, T. Inagaki, T. Ikuta, Y. Yamada, T. Kazama, K. Enbutsu, T. Umeki, R. Kasahara *et al.*, “100,000-spin coherent Ising machine,” *Science advances*, vol. 7, no. 40, p. eabh0952, 2021.

[4] C. Gambella and A. Simonetto, “Multiblock ADMM heuristics for mixed-binary optimization on classical and quantum computers,” *IEEE Transactions on Quantum Engineering*, vol. 1, pp. 1–22, 2020.

[5] C.-Y. Chang, E. Jones, Y. Yao, P. Graf, and R. Jain, “On Hybrid Quantum and Classical Computing Algorithms for Mixed-Integer Programming,” *arXiv e-prints*, pp. arXiv–2010, 2020.

[6] Z. Zhao, L. Fan, and Z. Han, “Hybrid Quantum Benders’ Decomposition For Mixed-integer Linear Programming,” *arXiv preprint arXiv:2112.07109*, 2021.

[7] N. G. Paterakis, “Hybrid Quantum-Classical Multi-cut Benders Approach with a Power System Application,” *arXiv preprint arXiv:2112.05643*, 2021.

[8] S. Burer, “On the copositive representation of binary and continuous nonconvex quadratic programs,” *Mathematical Programming*, vol. 120, no. 2, p. 479–495, 2009.

[9] A. Yurtsever, T. Birdal, and V. Golyanik, “Q-FW: A Hybrid Classical-Quantum Frank-Wolfe for Quadratic Binary Optimization,” *arXiv preprint arXiv:2203.12633*, 2022.

[10] D. Venturelli, D. Marchand, and G. Rojo, “Job shop scheduling solver based on quantum annealing,” in *Proc. of ICAPS-16 Workshop on Constraint Satisfaction Techniques for Planning and Scheduling (COPLAS)*, 2016, pp. 25–34.

[11] S. Harwood, C. Gambella, D. Trenev, A. Simonetto, D. Bernal, and D. Greenberg, “Formulating and solving routing problems on quantum computers,” *IEEE Transactions on Quantum Engineering*, vol. 2, pp. 1–17, 2021.

[12] C. F. Negre, H. Ushijima-Mwesigwa, and S. M. Mniszewski, “Detecting multiple communities using quantum annealing on the D-Wave system,” *Plos one*, vol. 15, no. 2, p. e0227538, 2020.

[13] A. Lucas, “Ising formulations of many NP problems,” *Frontiers in physics*, p. 5, 2014.

[14] A. Callison and N. Chancellor, “Hybrid quantum-classical algorithms in the noisy intermediate-scale quantum era and beyond,” *Physical Review A*, vol. 106, no. 1, p. 010101, 2022.

[15] S. Boyd, N. Parikh, E. Chu, B. Peleato, J. Eckstein *et al.*, “Distributed optimization and statistical learning via the alternating direction method of multipliers,” *Foundations and Trends® in Machine learning*, vol. 3, no. 1, pp. 1–122, 2011.

[16] S. Diamond, R. Takapoui, and S. Boyd, “A general system for heuristic minimization of convex functions over non-convex sets,” *Optimization Methods and Software*, vol. 33, no. 1, pp. 165–193, 2018.

[17] Y. Wang, W. Yin, and J. Zeng, “Global convergence of ADMM in nonconvex nonsmooth optimization,” *Journal of Scientific Computing*, vol. 78, no. 1, pp. 29–63, 2019.

[18] H. Attouch, J. Bolte, and B. F. Svaiter, “Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward–backward splitting, and regularized Gauss–Seidel methods,” *Mathematical Programming*, vol. 137, no. 1, pp. 91–129, 2013.

[19] M. Booth, S. Reinhardt, and A. Roy, “Partitioning Optimization Problems for Hybrid Classical/Quantum Execution,” D-Wave, Tech. Rep., 2017.

[20] F. Glover and S. Hanafi, “Tabu search and finite convergence,” *Discrete Applied Mathematics*, vol. 119, no. 1-2, pp. 3–36, 2002.

[21] H. Alghassi, R. Dridi, and S. Tayur, “Graver bases via quantum annealing with application to non-linear integer programs,” *arXiv preprint arXiv:1902.04215*, 2019.

[22] T. Albash and D. A. Lidar, “Adiabatic quantum computation,” *Reviews of Modern Physics*, vol. 90, no. 1, p. 015002, 2018.

[23] P. Hauke, H. G. Katzgraber, W. Lechner, H. Nishimori, and W. D. Oliver, “Perspectives of quantum annealing: Methods and implementations,” *Reports on Progress in Physics*, vol. 83, no. 5, p. 054401, 2020.

[24] M. Cerezo, A. Arrasmith, R. Babbush, S. C. Benjamin, S. Endo, K. Fujii, J. R. McClean, K. Mitarai, X. Yuan, L. Cincio *et al.*, “Variational quantum algorithms,” *Nature Reviews Physics*, vol. 3, no. 9, pp. 625–644, 2021.

[25] J. R. McClean, J. Romero, R. Babbush, and A. Aspuru-Guzik, “The theory of variational hybrid quantum-classical algorithms,” *New Journal of Physics*, vol. 18, no. 2, p. 023023, 2016.

[26] L. Bittel and M. Kliesch, “Training variational quantum algorithms is np-hard,” *Physical review letters*, vol. 127, no. 12, p. 120502, 2021.

[27] E. Farhi and A. W. Harrow, "Quantum supremacy through the quantum approximate optimization algorithm," *arXiv preprint arXiv:1602.07674*, 2016.

[28] J. Preskill, "Quantum computing in the nisq era and beyond," *Quantum*, vol. 2, p. 79, 2018.

[29] A. Uvarov and J. D. Biamonte, "On barren plateaus and cost function locality in variational quantum algorithms," *Journal of Physics A: Mathematical and Theoretical*, vol. 54, no. 24, p. 245301, 2021.

[30] M. Willsch, D. Willsch, F. Jin, H. De Raedt, and K. Michielsen, "Benchmarking the quantum approximate optimization algorithm," *Quantum Information Processing*, vol. 19, no. 7, pp. 1–24, 2020.

[31] M. P. Harrigan, K. J. Sung, M. Neeley, K. J. Satzinger, F. Arute, K. Arya, J. Atalaya, J. C. Bardin, R. Barends, S. Boixo *et al.*, "Quantum approximate optimization of non-planar graph problems on a planar superconducting processor," *Nature Physics*, vol. 17, no. 3, pp. 332–336, 2021.

[32] Y. R. Sanders, D. W. Berry, P. C. Costa, L. W. Tessler, N. Wiebe, C. Gidney, H. Neven, and R. Babbush, "Compilation of fault-tolerant quantum heuristics for combinatorial optimization," *PRX Quantum*, vol. 1, no. 2, p. 020312, 2020.

[33] Y. Yamamoto, K. Aihara, T. Leleu, K.-i. Kawarabayashi, S. Kako, M. Fejer, K. Inoue, and H. Takesue, "Coherent Ising machines—Optical neural networks operating at the quantum limit," *npj Quantum Information*, vol. 3, no. 1, pp. 1–15, 2017.

[34] N. Mohseni, P. L. McMahon, and T. Byrnes, "Ising machines as hardware solvers of combinatorial optimization problems," *Nature Reviews Physics*, pp. 1–17, 2022.

[35] Y. Shim, A. Jaiswal, and K. Roy, "Ising computation based combinatorial optimization using spin-hall effect (she) induced stochastic magnetization reversal," *Journal of Applied Physics*, vol. 121, no. 19, p. 193902, 2017.

[36] D. Pierangeli, G. Marcucci, and C. Conti, "Large-scale photonic ising machine by spatial light modulation," *Physical review letters*, vol. 122, no. 21, p. 213902, 2019.

[37] M. N. Bojnordi and E. Ipek, "Memristive boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning," in *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 2016, pp. 1–13.

[38] S. Matsubara, M. Takatsu, T. Miyazawa, T. Shibasaki, Y. Watanabe, K. Takemoto, and H. Tamura, "Digital annealer for high-speed solving of combinatorial optimization problems and its applications," in *2020 25th Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2020, pp. 667–672.

[39] P. L. McMahon, A. Marandi, Y. Haribara, R. Hamerly, C. Langrock, S. Tamate, T. Inagaki, H. Takesue, S. Utsunomiya, K. Aihara *et al.*, "A fully programmable 100-spin coherent ising machine with all-to-all connections," *Science*, vol. 354, no. 6312, pp. 614–617, 2016.

[40] J. Chou, S. Bramhavar, S. Ghosh, and W. Herzog, "Analog coupled oscillator based weighted ising machine," *Scientific reports*, vol. 9, no. 1, p. 14786, 2019.

[41] T. Albash and D. A. Lidar, "Demonstration of a scaling advantage for a quantum annealer over simulated annealing," *Physical Review X*, vol. 8, no. 3, p. 031016, 2018.

[42] L. Henriet, L. Beguin, A. Signoles, T. Lahaye, A. Browaeys, G.-O. Reymond, and C. Jurczak, "Quantum computing with neutral atoms," *Quantum*, vol. 4, p. 327, 2020.

[43] S. Boyd and L. Vandenberghe, "Convex Optimization," 2004.

[44] K. G. Murty and S. N. Kabadi, "Some NP-complete problems in quadratic and nonlinear programming," *Mathematical Programming*, vol. 39, no. 2, pp. 117–129, 1987.

[45] P. A. Parrilo, "Structured Semidefinite Programs and Semialgebraic Geometry Methods in Robustness and Optimization," Ph.D. dissertation, 2000.

[46] J. B. Lasserre, "Global optimization with polynomials and the problem of moments," *SIAM Journal on optimization*, vol. 11, no. 3, pp. 796–817, 2001.

[47] M. Dür and F. Rendl, "Conic optimization: a survey with special focus on copositive optimization and binary quadratic problems," *EURO Journal on Computational Optimization*, vol. 9, p. 100021, 2021.

[48] S. Burer, "Copositive programming," in *Handbook on semidefinite, conic and polynomial optimization.* Springer, 2012, pp. 201–218.

[49] M. Dür, "Copositive programming–a survey," in *Recent advances in optimization and its applications in engineering.* Springer, 2010, pp. 3–20.

[50] S. Boyd and L. Vandenberghe, "Localization and cutting-plane methods," *From Stanford EE 364b lecture notes*, 2007.

[51] K. K. Sivaramakrishnan and J. E. Mitchell, "Properties of a cutting plane method for semidefinite programming," *Pacific Journal of Optimization*, vol. 8, pp. 779–802, 2007.

[52] L. A. Rademacher, "Approximating the centroid is hard," in *Proceedings of the twenty-third annual symposium on Computational geometry*, 2007, pp. 302–305.

[53] A. Y. Levin, "An algorithm for minimizing convex functions," in *Doklady Akademii Nauk*, vol. 160, no. 6. Russian Academy of Sciences, 1965, pp. 1244–1247.

[54] N. Z. Shor, "Cut-off method with space extension in convex programming problems," *Cybernetics*, vol. 13, no. 1, pp. 94–96, 1977.

[55] D. B. Yudin and A. S. Nemirovski, "Evaluation of the information complexity of mathematical programming problems," *Ekonomika i Matematicheskie Metody*, vol. 12, pp. 128–142, 1976.

[56] L. G. Khachiyan, "Polynomial algorithms in linear programming," *USSR Computational Mathematics and Mathematical Physics*, vol. 20, no. 1, pp. 53–72, 1980.

[57] L. G. Khachiyan, S. P. Tarasov, and I. Erlikh, "The method of inscribed ellipsoids," in *Soviet Math. Dokl*, vol. 37, no. 1, 1988, pp. 226–230.

[58] Y. Nesterov and A. Nemirovski, "Self-concordant functions and polynomial time methods in convex programming," *USSR Academy of Sciences, Central Economic&Mathematical Institute, Moscow*, 1989.

[59] P. M. Vaidya, "A new algorithm for minimizing convex functions over convex sets," in *30th Annual Symposium on Foundations of Computer Science.* IEEE Computer Society, 1989, pp. 338–343.

[60] D. S. Atkinson and P. M. Vaidya, "A cutting plane algorithm for convex programming that uses analytic centers," *Mathematical Programming*, vol. 69, no. 1, pp. 1–43, 1995.

[61] D. Bertsimas and S. Vempala, "Solving convex programs by random walks," *Journal of the ACM (JACM)*, vol. 51,

no. 4, pp. 540–556, 2004.

[62] Y. T. Lee, A. Sidford, and S. C.-w. 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*. Ieee, 2015, pp. 1049–1065.

[63] C. Guo, M. Bodur, and J. A. Taylor, “Copositive duality for discrete markets and games,” *arXiv preprint arXiv:2101.05379*, 2021.

[64] R. Badenbroek and E. de Klerk, “An analytic center cutting plane method to determine complete positivity of a matrix,” *INFORMS Journal on Computing*, vol. 34, no. 2, pp. 1115–1125, 2022.

[65] K. M. Anstreicher, “Testing copositivity via mixed–integer linear programming,” *Linear Algebra and its Applications*, vol. 609, pp. 218–230, 2021.

[66] M. Dür and J.-B. Hiriart-Urruty, “Testing copositivity with the help of difference-of-convex optimization,” *Mathematical Programming*, vol. 140, no. 1, pp. 31–43, 2013.

[67] J.-B. Hiriart-Urruty and A. Seeger, “A variational approach to copositive matrices,” *SIAM review*, vol. 52, no. 4, pp. 593–629, 2010.

[68] C. Brás, G. Eichfelder, and J. Júdice, “Copositivity tests based on the linear complementarity problem,” *Computational Optimization and Applications*, vol. 63, no. 2, pp. 461–493, 2016.

[69] W. Xia, J. C. Vera, and L. F. Zuluaga, “Globally solving nonconvex quadratic programs via linear integer programming techniques,” *INFORMS Journal on Computing*, vol. 32, no. 1, pp. 40–56, 2020.

[70] F. Khosravi, U. Yildiz, A. Scherer, and P. Ronagh, “Non-convex quadratic programming using coherent optical networks,” *arXiv preprint arXiv:2209.04415*, 2022.

[71] E. De Klerk and D. V. Pasechnik, “Approximation of the stability number of a graph via copositive programming,” *SIAM Journal on Optimization*, vol. 12, no. 4, pp. 875–892, 2002.

[72] *Gurobi Optimizer Reference Manual*, 2022, available at http://www.gurobi.com.

[73] *dwave-neal Documentation*, D-Wave Systems Inc, 2021, available at https://docs.ocean.dwavesys.com/_/downloads/neal/en/latest/pdf/.

[74] T. F. Rønnow, Z. Wang, J. Job, S. Boixo, S. V. Isakov, D. Wecker, J. M. Martinis, D. A. Lidar, and M. Troyer, “Defining and detecting quantum speedup,” *science*, vol. 345, no. 6195, pp. 420–424, 2014.

[75] J. King, S. Yarkoni, M. M. Nevisi, J. P. Hilton, and C. C. McGeoch, “Benchmarking a quantum annealing processor with the time-to-target metric,” *arXiv preprint arXiv:1508.05087*, 2015.

[76] R. Quintero, D. Bernal, T. Terlaky, and L. F. Zuluaga, “Characterization of QUBO reformulations for the maximum k-colorable subgraph problem,” *Quantum Information Processing*, vol. 21, no. 3, pp. 1–36, 2022.

[77] D. Bertsekas, *Convex optimization theory*. Athena Scientific, 2009, vol. 1.

[78] S. Kim and M. Kojima, “Strong duality of a conic optimization problem with two cones and a single equality constraint,” *arXiv preprint arXiv:2111.03251*, 2021.

[79] D. Cifuentes, S. S. Dey, and J. Xu, “Sensitivity analysis for mixed binary quadratic programming,” *arXiv preprint arXiv:2312.06714*, 2023.

[80] S. Karimi and P. Ronagh, “Practical integer-to-binary mapping for quantum annealers,” *Quantum Information Processing*, vol. 18, no. 4, pp. 1–24, 2019.

[81] J. Bergstra, D. Yamins, and D. Cox, “Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures,” in *International conference on machine learning*. Pmlr, 2013, pp. 115–123.

6. Appendix.

6.1. Proof of strong duality. Problem (CPP) is equivalent to the following homogenous form completely positive program (i.e., min (CPP) = min (Hom-CPP)):

$$\text{(Hom-CPP)}\qquad \begin{array}{cl} \underset{x \in \mathbb{R}^n, X \in \mathbb{R}^{n\times n}}{\text{minimize}} & \left\langle \begin{pmatrix} Q & c \\ c^\top & \cdot \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle \\ \text{subject to} & \left\langle \begin{pmatrix} A_{i,*}^\top A_{i,*} & -b_i A_{i,*}^\top \\ -b_i A_{i,*} & b_i^2 \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle = 0, \\ & \left\langle \begin{pmatrix} -\mathbb{1}_{\{j\}} \mathbb{1}_{\{j\}}^\top & \frac{1}{2}\mathbb{1}_{\{j\}} \\ \frac{1}{2}\mathbb{1}_{\{j\}}^\top & \cdot \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle = 0, \; \forall j \in B, \\ & \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \in \mathcal{C}_{n+1}^*. \end{array}$$

This form will be useful for proving strong duality. Because the homogenized form of the equality constraints form a cone, this perspective will help prove strong duality between Problem (CPP) and its dual. The Lagrangian dual of (Hom-CPP) is the following copositive optimization problem:

$$\text{(Hom-COP)}\qquad \begin{array}{cl} \underset{\mu, \lambda, \gamma}{\text{maximize}} & \gamma \\ \text{subject to} & \hat{M}(\mu, \lambda, \gamma) \in \mathcal{C}_{n+1}, \end{array}$$

where $\hat{M}(\mu, \lambda, \gamma)$ is defined as

$$\begin{aligned} \hat{M}(\mu, \lambda, \gamma) := & \begin{pmatrix} Q & c \\ c^\top & \cdot \end{pmatrix} - \sum_i \mu_i \begin{pmatrix} A_{i,*}^\top A_{i,*} & -b_i A_{i,*}^\top \\ -b_i A_{i,*} & b_i^2 \end{pmatrix} \\ & - \sum_{j\in B} \lambda_j \begin{pmatrix} -\mathbb{1}_{\{j\}} \mathbb{1}_{\{j\}}^\top & \frac{1}{2}\mathbb{1}_{\{j\}} \\ \frac{1}{2}\mathbb{1}_{\{j\}}^\top & \cdot \end{pmatrix} - \gamma \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix}. \end{aligned} \tag{6.1}$$

Theorem 6.1 (Homogeneous Strong Duality). *If Problem* (MBQP) *is feasible with bounded feasible region, then strong duality holds between Problems* (Hom-CPP) *and* (Hom-COP)*. Moreover, an ϵ optimal value of* (Hom-COP) *is obtained by a feasible solution, where $\epsilon > 0$ can be arbitrarily small.*

Proof. Notice that the set of affine constraints,

$$\mathcal{NULL} := \left\{ \tilde{X} \mid \left\langle \begin{pmatrix} A_{i,*}^\top A_{i,*} & -b_i A_{i,*}^\top \\ -b_i A_{i,*} & b_i^2 \end{pmatrix}, \tilde{X} \right\rangle = 0, \left\langle \begin{pmatrix} -\mathbb{1}_{\{j\}} \mathbb{1}_{\{j\}}^\top & \frac{1}{2}\mathbb{1}_{\{j\}} \\ \frac{1}{2}\mathbb{1}_{\{j\}}^\top & \cdot \end{pmatrix}, \tilde{X} \right\rangle = 0 \right\}, \tag{6.2}$$

forms a cone. So, we could express (Hom-CPP) as the following optimization problem:

$$\begin{aligned} \underset{\tilde{X} \in \mathbb{R}^{(n+1)\times(n+1)}}{\text{minimize}} \quad & \left\langle \begin{pmatrix} Q & c \\ c^\top & \cdot \end{pmatrix}, \tilde{X} \right\rangle \\ \text{subject to} \quad & \left\langle \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix}, \tilde{X} \right\rangle = 1, \\ & \tilde{X} \in \mathcal{C}_{n+1}^* \cap \mathcal{NULL}. \end{aligned} \tag{6.3}$$

As a quick aside, rewriting the problem in this way does not change the Lagrangian dual problem. To see this, we first write the Lagrangian dual of Problem (6.3) as

$$\begin{aligned} \underset{\gamma \in \mathbb{R}, \tilde{M} \in \mathbb{R}^{(n+1)\times(n)}}{\text{maximize}} \quad & \gamma \\ \text{subject to} \quad & \tilde{M} = \begin{pmatrix} Q & c \\ c^\top & \cdot \end{pmatrix} - \gamma \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix}, \\ & \tilde{M} \in \mathcal{C}_{n+1} + \mathcal{NULL}^* \end{aligned} \tag{6.4}$$

and notice that $\mathcal{NULL}^*$ is spanned by

$$\left\{ \begin{pmatrix} A_{i,*}^\top A_{i,*} & -b_i A_{i,*}^\top \\ -b_i A_{i,*} & b_i^2 \end{pmatrix} \right\} \cup \left\{ \begin{pmatrix} -\mathbb{1}_{\{j\}} \mathbb{1}_{\{j\}}^\top & \frac{1}{2}\mathbb{1}_{\{j\}} \\ \frac{1}{2}\mathbb{1}_{\{j\}}^\top & \cdot \end{pmatrix}] \right\}. \tag{6.5}$$

Here, we take care to note that the *Lagrangian* dual may differ from the *conic* dual. In particular, feasibility of (MBQP) is sufficient for ensuring strong duality when $\tilde{M}$ is optimized over $(C^* \cap \mathcal{NULL})^*$ [77, Prop 5.3.9]; however, it is not guaranteed that $(C^* \cap \mathcal{NULL})^*$ is equal to $C + \mathcal{NULL}^*$; this is the case if and only if $C + \mathcal{NULL}^*$ is closed.

To establish strong duality, we will first assert that if Problem (MBQP) is feasible with a bounded feasible region, then Problem (Hom-CPP) has a nonempty and bounded set of optimal solutions. This follows directly from [8, Corollary 2.6], which states that for all optimal solutions, (x^*, X^*), of (Hom-CPP), x^* must lie within the convex hull of optimal solutions for (MBQP). If the set of optimal solutions for (MBQP) is non-empty and bounded, so is their convex hull, proving the boundedness of x^*. Moreover, because the optimal solution may be factored in the form

$$\begin{pmatrix} X^* & x^* \\ x^{*\top} & 1 \end{pmatrix} = \sum_k \begin{pmatrix} x^{(k)} \\ \xi^{(k)} \end{pmatrix} \begin{pmatrix} x^{(k),\top} & \xi^{(k)} \end{pmatrix} \tag{6.6}$$

with $\xi^{(k)} > 0$ and $\sum_k (\xi^{(k)})^2 = 1$ (by definition of copositivity), X^* can be expressed as the sum of the outer products of optimal solutions of (MBQP) with themselves. In other words, because each $x^{(k)}$ is bounded, $X^* = \sum_k x^{(k)} x^{(k),\top}$ must be bounded as well. This establishes that the set of optimal solutions of (Hom-CPP) is also non-empty and bounded, allowing us to apply [78, Theorem 1.1], which establishes strong duality of conic optimization problems with two cone constraints and a single hyperplane constraint under non-emptiness and boundedness of the optimal solution set. The establishes that

$$\max\,(\text{Hom-COP}) = \min\,(\text{Hom-CPP}). \tag{6.7}$$

While the optimal objective of (Hom-COP) may not be exactly attainable, there exists a feasible solution with objective value $\max\,(\text{Hom-COP}) - \epsilon$ where $\epsilon > 0$ can be arbitrarily small [79]. This is not restrictive, as most numerical solvers only compute optimums with finite precision. ∎

While Theorem 6.1 establishes strong duality of the homogenous CPP, we have yet to show that the non-homogeneous form also exhibits strong duality. In order to do so, we will show that the supremum of the (non-homogeneous) copositive program upper-bounds that of the homogeneous program.

Theorem 6.2 (Inhomogeneous Lower Bound). *The optimal objective of Problem* (COP) *is at least that of Problem* (Hom-COP) *(i.e.,* max (COP) $\geq$ max (Hom-COP)*).*

Proof. We will do this by showing that for each $(\hat{\mu}, \hat{\lambda}, \hat{\gamma})$ there exists (μ, λ, γ) such that

$$M(\mu, \lambda, \gamma) = \hat{M}(\hat{\mu}, \hat{\lambda}, \hat{\gamma}), \tag{6.8}$$

and

$$\gamma + \sum_i \mu_i^{(\text{lin})} b_i + \mu_i^{(\text{quad})} b_i^2 = \hat{\gamma}. \tag{6.9}$$

In other words, any feasible solution for (Hom-COP) can be transformed into a feasible solution for (COP) with equal objective value. To see this, we will suggestively break up $\gamma = \gamma^{(\text{res})} + \sum_i \gamma_i$ so equation (3.1) can be expanded as

$$\begin{aligned} M(\mu, \lambda, \gamma) = & \begin{pmatrix} Q & c \\ c^\top & \cdot \end{pmatrix} \\ & - \sum_i \left(\mu_i^{(\text{lin})} \begin{pmatrix} \cdot & \frac{1}{2} A_{i,*}^\top \\ \frac{1}{2} A_{i,*} & \cdot \end{pmatrix} + \mu_i^{(\text{quad})} \begin{pmatrix} A_{i,*}^\top A_{i,*} & \cdot \\ \cdot & \cdot \end{pmatrix} + \gamma_i \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix} \right) \\ & - \sum_{j \in B} \lambda_j \begin{pmatrix} -\mathbb{1}_{\{j\}} \mathbb{1}_{\{j\}}^\top & \frac{1}{2} \mathbb{1}_{\{j\}} \\ \frac{1}{2} \mathbb{1}_{\{j\}}^\top & \cdot \end{pmatrix} - \gamma^{(\text{res})} \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix} \end{aligned} \tag{6.10}$$

Then, the proposed (μ, λ, γ) is given by

$$\lambda_j = \hat{\lambda}_j \tag{6.11}$$

$$\mu_i^{(\text{lin})} = -2 b_i \hat{\mu}_i \tag{6.12}$$

$$\mu_i^{(\text{quad})} = \hat{\mu}_i \tag{6.13}$$

$$\gamma_i = b_i^2 \hat{\mu}_i \tag{6.14}$$

$$\gamma^{(\text{res})} = \hat{\gamma} \tag{6.15}$$

Then, notice that

$$\mu_i^{(\text{lin})} \begin{pmatrix} \cdot & \frac{1}{2} A_{i,*}^\top \\ \frac{1}{2} A_{i,*} & \cdot \end{pmatrix} + \mu_i^{(\text{quad})} \begin{pmatrix} A_{i,*}^\top A_{i,*} & \cdot \\ \cdot & \cdot \end{pmatrix} + \gamma_i \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix} \tag{6.16}$$

$$= -2 b_i \hat{\mu}_i \begin{pmatrix} \cdot & \frac{1}{2} A_{i,*}^\top \\ \frac{1}{2} A_{i,*} & \cdot \end{pmatrix} + \hat{\mu}_i \begin{pmatrix} A_{i,*}^\top A_{i,*} & \cdot \\ \cdot & \cdot \end{pmatrix} + b_i^2 \hat{\mu}_i \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix} \tag{6.17}$$

$$= \hat{\mu}_i \left(-2 b_i \begin{pmatrix} \cdot & \frac{1}{2} A_{i,*}^\top \\ \frac{1}{2} A_{i,*} & \cdot \end{pmatrix} + \begin{pmatrix} A_{i,*}^\top A_{i,*} & \cdot \\ \cdot & \cdot \end{pmatrix} + b_i^2 \begin{pmatrix} \cdot & \cdot \\ \cdot & 1 \end{pmatrix} \right) \tag{6.18}$$

$$= \hat{\mu}_i \begin{pmatrix} A_{i,*}^\top A_{i,*} & -b_i A_{i,*}^\top \\ -b_i A_{i,*} & b_i^2 \end{pmatrix} \tag{6.19}$$

so by matching up terms in the sums, we see that $M(\mu, \lambda, \gamma) = \hat{M}(\hat{\mu}, \hat{\lambda}, \hat{\gamma})$. As for the objective value,

$$\gamma + \sum_i \mu_i^{(\text{lin})} b_i + \mu_i^{(\text{quad})} b_i^2 \tag{6.20}$$

$$= \gamma^{(\text{res})} + \sum_i \mu_i^{(\text{lin})} b_i + \mu_i^{(\text{quad})} b_i^2 + \gamma_i \tag{6.21}$$

$$= \hat{\gamma} + \sum_i -2 b_i^2 \hat{\mu}_i + b_i^2 \hat{\mu}_i + b_i^2 \hat{\mu}_i \tag{6.22}$$

$$= \hat{\gamma} + \sum_i \hat{\mu}_i (-2 b_i^2 + b_i^2 + b_i^2) \tag{6.23}$$

$$= \hat{\gamma} \tag{6.24}$$

so for each $(\hat{\mu}, \hat{\lambda}, \hat{\gamma})$ the proposed (μ, λ, γ) has equal objective value. □

Corollary 6.3. *If Problem* (MBQP) *is feasible with bounded feasible region, then strong duality holds between Problems* (CPP) *and* (COP)*.*

Proof. Theorem shows that max (COP) $\geq$ max (Hom-COP). So we have max (Hom-COP) $\leq$ max (COP) $\leq$ min (CPP) $=$ min (Hom-CPP). Combining this with max (Hom-COP) $=$ min (Hom-CPP) we get

$$\max(\text{Hom-COP}) = \max(\text{COP}) = \min(\text{CPP}) = \min(\text{Hom-CPP}). \tag{6.25}$$

Thus, strong duality must hold between (COP) and (CPP). □

6.2. Discretizing the copositivity checks.

6.2.1. Constructing the QUBO. In this section, we will discuss forming the QUBO to approximate the copositivity checks. Formally, instead of solving (3.8) with feasible region $\{z \in \mathbb{R}^{n+1}_{\geq 0} \mid \|z\|_\infty \leq 1\}$, we will approximate the feasible region with $\{0, \frac{1}{K}, \dots, \frac{K-1}{K}, 1\}^{n+1}$, leading to a quadratic unconstrained integer optimization,

$$\text{(QUIO)} \qquad \begin{aligned} \underset{z}{\text{minimize}} \quad & z^\top M(\mu, \lambda, \gamma) z \\ \text{subject to} \quad & z \in \left\{0, \frac{1}{K}, \dots, \frac{K-1}{K}, 1\right\}^{n+1} \end{aligned}$$

For simplicity, assume that $K = 2^k - 1$ for some $k \in \mathbb{Z}_{>0}$. Then Problem (QUIO) is equivalent to minimizing (QUBO), where

$$\hat{M}(\mu, \lambda, \gamma) = \mathcal{D}^\top M(\mu, \lambda, \gamma) \mathcal{D} \tag{6.26}$$

and

$$\mathcal{D} := \frac{1}{K} \begin{pmatrix} 2^0 & \cdots & 2^{k-1} & 0 & \cdots & 0 & \cdots & 0 & \cdots & 0 \\ 0 & \cdots & 0 & 2^0 & \cdots & 2^{k-1} & \cdots & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \cdots & 0 & 0 & \cdots & 0 & \cdots & 2^0 & \cdots & 2^{k-1} \end{pmatrix}, \tag{6.27}$$

over the variables $\hat{z} \in \{0,1\}^{k(n+1)}$. The new variables $\hat{z}$ simply represent the binary expansion of z, i.e., $z = \frac{1}{K}\mathcal{D}\hat{z}$. One could also use a unary expansion at the expense of a larger size expansion and redundancy in the encoding. Additionally, while we have written out a uniform expansion for all variables, it is possible to have a heterogenous discretization scheme. More sophisticated discretization schemes are discussed in depth in [80].

6.2.2. Choosing a discretization size. When discretizing the copositivity checks, it is critical to ensure that the discretization size is fine enough. This section provides guidance for choosing a discretization size given a particular QUBO. To do so, we will consider minimizing $z^\top M z$ over a discrete grid $z \in \{0, \frac{1}{K}, \dots, \frac{K-1}{K}, 1\}^n$ and bound the difference if we have minimized $\hat{z}^\top M \hat{z}$ over the hypercube instead, $\hat{z} \in [0,1]^n$. In particular, we are interested in the case where there are no certificates of non-copositivity on the discrete grid, i.e., $z^\top M z \geq 0$ for all $z \in \{0, \frac{1}{K}, \dots, \frac{K-1}{K}, 1\}^n$, yet there is $\hat{z} \in [0,1]^n$ with $\hat{z}^\top M \hat{z} = -\delta < 0$. We will decompose $\hat{z} = z + \Delta$ as the nearest grid point, $z \in \{0, \frac{1}{K}, \dots, \frac{K-1}{K}, 1\}^n$, plus a small correction factor $\Delta \in \mathbb{R}^n$.

Because the norm of this correction factor is bounded by $\|\Delta\|_\infty \leq \frac{1}{2K}$ (i.e., by rounding), we will lower-bound $(z+\Delta)^\top M (z+\Delta)$ as a function of $\|\Delta\|$. Expanding $(z+\Delta)^\top M(z+\Delta)$ out we get

$$(z+\Delta)^\top M(z+\Delta) = z^\top M z + 2\Delta^\top M z + \Delta^\top M \Delta. \tag{6.28}$$

Applying the assumption that $z^\top M z \geq 0$ for all $z \in \{0, \frac{1}{K}, \dots, \frac{K-1}{K}, 1\}^n$, we get

$$(z+\Delta)^\top M(z+\Delta) \geq -|2\Delta^\top M z + \Delta^\top M \Delta|. \tag{6.29}$$

In order to lower-bound, $\hat{z}^\top M \hat{z}$, we upper-bound $|2\Delta^\top M z + \Delta^\top M \Delta|$,

$$\begin{aligned} |2\Delta^\top M z + \Delta^\top M \Delta| &= \left\|2\Delta^\top M z + \Delta^\top M \Delta\right\|_\infty && (6.30) \\ &\leq 2\|\Delta\|_\infty \|z\|_\infty \|M\|_\infty + \|\Delta\|_\infty^2 \|M\|_\infty && (6.31) \\ &\leq (2\|\Delta\|_\infty + \|\Delta\|_\infty^2)\|M\|_\infty && (6.32) \\ &\leq \left(\frac{1}{K} + \frac{1}{4K^2}\right)\|M\|_\infty. && (6.33) \end{aligned}$$

Recall that we assume the minimum copositivity check is achieved at $-\delta = (z+\Delta)^\top M(z+\Delta)$, so if $K > \frac{1}{2(\sqrt{\frac{\delta}{\|M\|_\infty}+1}-1)}$, then there exists $z \in \{0, \frac{1}{K}, \dots, \frac{K-1}{K}, 1\}^n$ with $z^\top M z < 0$. This represents the coarsest discretization where optimizing over the discrete grid rather than the unit hypercube is insufficient for detecting the certificate of non-copositivity.

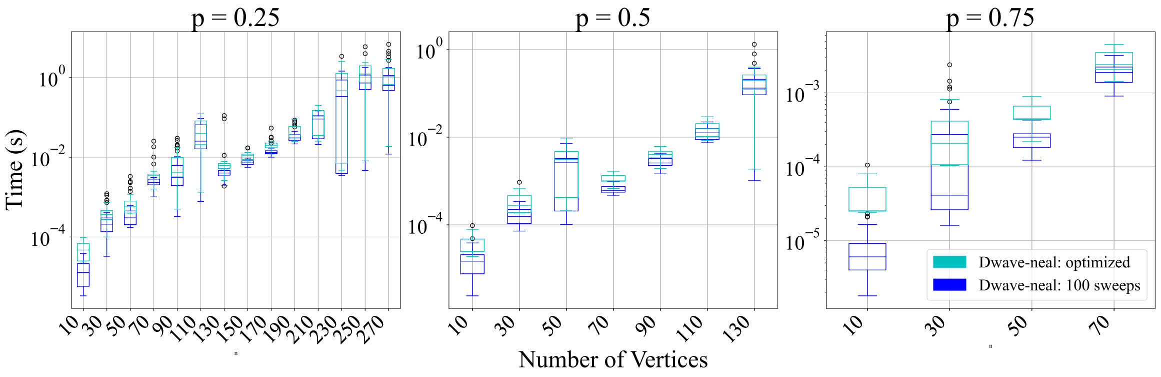


Fig. 8: This figure plots the optimized $\mathtt{TTT}_{0.999}$ when `Neal` was run with 100 sweeps. Optimization produces an order of magnitude speed-up for graphs with 10 nodes but does not result in significant speed-ups for larger graphs.

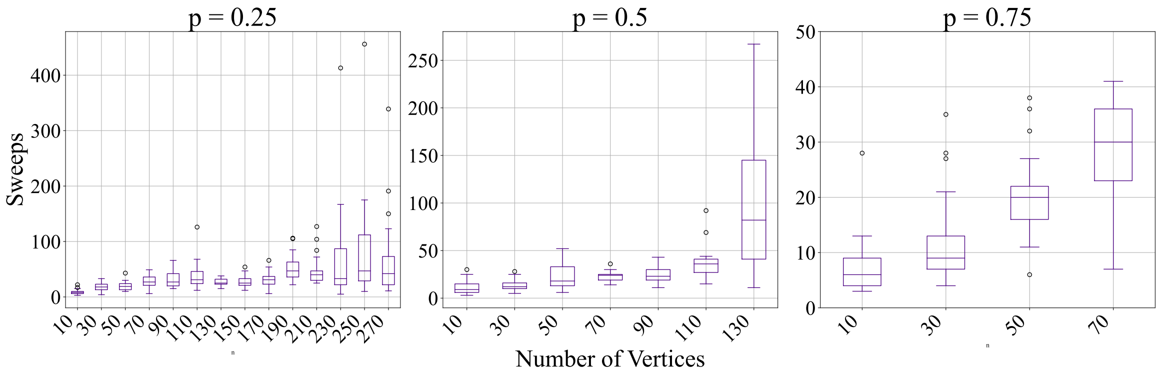


Fig. 9: This figure plots the optimal number of sweeps for each of the problem instances. The optimal number of sweeps increases with the graph size. However, graphs of densities $p = 0.25$ and $p = 0.75$ require a comparable number of sweeps for graphs of the same size, while fewer sweeps are required for graphs with density $p = 0.5$

6.3. Hyper-parameter optimization. To investigate further speed-ups from turning the simulated annealing parameters, we optimized the number of sweeps using `Hyperopt` [81] with 25 trials for each instance. Figure 8 plots the optimized $\mathtt{TTT}_{0.99}$ and the $\mathtt{TTT}_{0.99}$ when `Neal` was run with 100 sweeps. While the optimization produced significant relative improvements for graphs with 10 nodes, the improvement for larger graphs remained marginal, especially in light of the computational overhead required to optimize the parameters. Figure 9 plots the optimal number of sweeps for each problem instance. Generally, the optimal number of sweeps increases with the number of vertices. While graphs of densities $p = 0.25$ and $p = 0.75$ require a comparable number of sweeps for graphs of the same size, fewer sweeps are required for graphs with density $p = 0.5$.

6.4. Illustrative example. In this section, we will walk through a small MBQP to illustrate the translation into the equivalent copositive program. Consider the following mixed-binary optimization problem:

$$\text{(Ex-MBQP)} \qquad \begin{aligned} \underset{x_1, x_2}{\text{minimize}} \quad & \begin{pmatrix} x_1 & x_2 \end{pmatrix} \begin{pmatrix} 1 & -1 \\ -1 & \cdot \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ \text{subject to} \quad & x_1 + x_2 = 1, \\ & x_1,\, x_2 \in \mathbb{R}_{\geq 0}. \end{aligned}$$

The optimal solution is given by

$$x_1^* = \frac{1}{3}, \quad x_2^* = \frac{2}{3}, \tag{6.34}$$

which gives an optimal objective value of $-\frac{1}{3}$.

The equivalent completely positive program is given by

$$
\text{(Ex-CPP)} \qquad
\begin{aligned}
\underset{X\in\mathbb{R}^{2\times 2},\, x\in\mathbb{R}^2}{\text{minimize}} \quad & \left\langle \begin{pmatrix} 1 & -1 & \cdot \\ -1 & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle \\
\text{subject to} \quad & \left\langle \begin{pmatrix} \cdot & \cdot & 1 \\ \cdot & \cdot & 1 \\ 1 & 1 & \cdot \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle = 2, \\
& \left\langle \begin{pmatrix} 1 & 1 & \cdot \\ 1 & 1 & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle = 1, \\
& \left\langle \begin{pmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & 1 \end{pmatrix}, \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \right\rangle = 1, \\
& \begin{pmatrix} X & x \\ x^\top & 1 \end{pmatrix} \in \mathcal{C}_3^*.
\end{aligned}
$$

The optimal solution of (Ex-CPP) is determined by the quadratic expansion of the optimal solution of (Ex-MBQP) as follows:

$$
\begin{pmatrix} X^* & x^* \\ x^{*\top} & 1 \end{pmatrix} = \begin{pmatrix} x_1^* \\ x_2^* \\ 1 \end{pmatrix} \begin{pmatrix} x_1^* & x_2^* & 1 \end{pmatrix} = \begin{pmatrix} 1/9 & 2/9 & 1/3 \\ 2/9 & 4/9 & 2/3 \\ 1/3 & 2/3 & 1 \end{pmatrix} \tag{6.35}
$$

The dual copositive program is given by

$$
\text{(Ex-COP)} \qquad
\begin{aligned}
\underset{\mu,\lambda,\gamma}{\text{maximize}} \quad & \gamma + 2\mu^{(\text{lin})} + \mu^{(\text{quad})} \\
\text{subject to} \quad & M(\mu,\lambda,\gamma) \in \mathcal{C}_3
\end{aligned}
$$

where

$$
M(\mu,\lambda,\gamma) = \begin{pmatrix} 1-\mu^{(\text{quad})} & -1-\mu^{(\text{quad})} & -\mu^{(\text{lin})} \\ -1-\mu^{(\text{quad})} & -\mu^{(\text{quad})} & -\mu^{(\text{lin})} \\ -\mu^{(\text{lin})} & -\mu^{(\text{lin})} & -\gamma \end{pmatrix} \tag{6.36}
$$

Figure 10 plots the outer bounding ellipsoid for the first 9 iterations of the copositive cutting plane algorithm (with the ellipsoid method as the cutting-plane algorithm) applied to Problem (Ex-COP). For each iteration, the red dot depicts the test point, and the blue ellipsoid plots the outer bounding ellipsoid at the start of the iteration. The initial ellipsoid is chosen to be a sphere, but as the algorithm progresses, we observe that the outer bounding ellipsoid becomes elongated. This behavior is explained by the fact that the optimal solution set for this particular problem is a line.

6.5. Ellipsoid Algorithm. In this section, we will overview the ellipsoid algorithm as a representative example of cutting-plane algorithms [50]. As suggested by its name, the outer approximation defined by an ellipsoid. An ellipsoid in $\mathbb{R}^m$ is parametrized by a center, $x\in\mathbb{R}^m$, and positive definite matrix, $P\in\mathcal{S}^m_{++}$, and is defined as

$$
\mathcal{E}(x,P) := \{s\in\mathbb{R}^m \mid (s-x)P^{-1}(s-x) \le 1\}. \tag{6.37}
$$

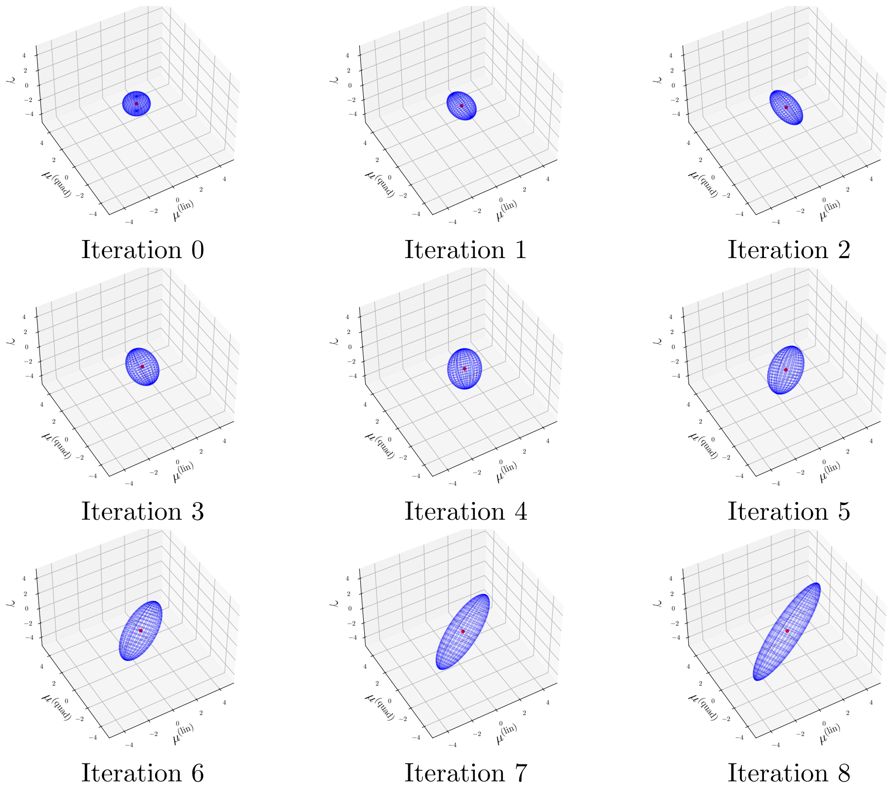
The volume of $\mathcal{E}(x,P)$ scales with the determinant of P,

$$
\texttt{Vol}(\mathcal{E}(x,P)) = \frac{\pi^{m/2}}{\Gamma\left(\frac{m}{2}+1\right)}\sqrt{\det(P)}. \tag{6.38}
$$

In the ellipsoid algorithm, the center will always be the test point. Given a separating hyperplane for x (recall that this is defined by a vector $a\in\mathbb{R}^m$ such that $a^\top s \le a^\top x$ for all $s\in S$), the ellipsoid updates the outer approximation with the minimum volume ellipsoid containing both $\mathcal{E}(x,P)$ and $\{s\in\mathbb{R}^m \mid a^\top s \le a^\top x\}$. Conveniently, this ellipsoid, $\mathcal{E}(\hat{x},\hat{P})$, has a closed form representation with

$$
\hat{x}(x,P,a) = x - \frac{Pa}{(m+1)\sqrt{a^\top Pa}}, \tag{6.39}
$$

$$
\hat{P}(x,P,a) = \frac{m^2}{m^2-1}\left(P - \frac{2Paa^\top P}{(m+1)a^\top Pa}\right). \tag{6.40}
$$



Iteration 0 Iteration 1 Iteration 2

Iteration 3 Iteration 4 Iteration 5

Iteration 6 Iteration 7 Iteration 8

Fig. 10: This figure plots the outer bounding ellipsoid for the first nine iterations of the copositive cutting plane algorithm applied to Problem (Ex-COP). In each plot, the red dot depicts the test point, and the blue ellipsoid plots the outer bounding ellipsoid at the start of the iteration.

Now we are in a position to present the ellipsoid algorithm in the terminology of Section 3.2. To initialize the algorithm, the user chooses x_0 and P_0 appropriately and chooses a final tolerance r. The outer approximations for each iteration are maintained via x_k and P_k, i.e., $S_k = \mathcal{E}(x_k, P_k)$. Evaluating the center of S_k involves simply returning the stored value for x_k, i.e., $\texttt{Center}(S_k) = x_k$ The initial volume is determined from Equation (6.38) as $R = \texttt{Vol}(\mathcal{E}(x_0, P_0))$. At each iteration, the separation oracle, $\texttt{Oracle}(x_k)$, is evaluated using Algorithm 2, and outer approximation is updated as follows:

$$x_{k+1} = \hat{x}(x_k, P_k, \texttt{Oracle}(x_k)), \tag{6.41}$$

$$P_{k+1} = \hat{P}(x_k, P_k, \texttt{Oracle}(x_k)), \tag{6.42}$$

$$\texttt{Add\_Cut}(S_k, \texttt{Oracle}(x)) = \mathcal{E}(x_{k+1}, P_{k+1}). \tag{6.43}$$

6.6. Additional plots.

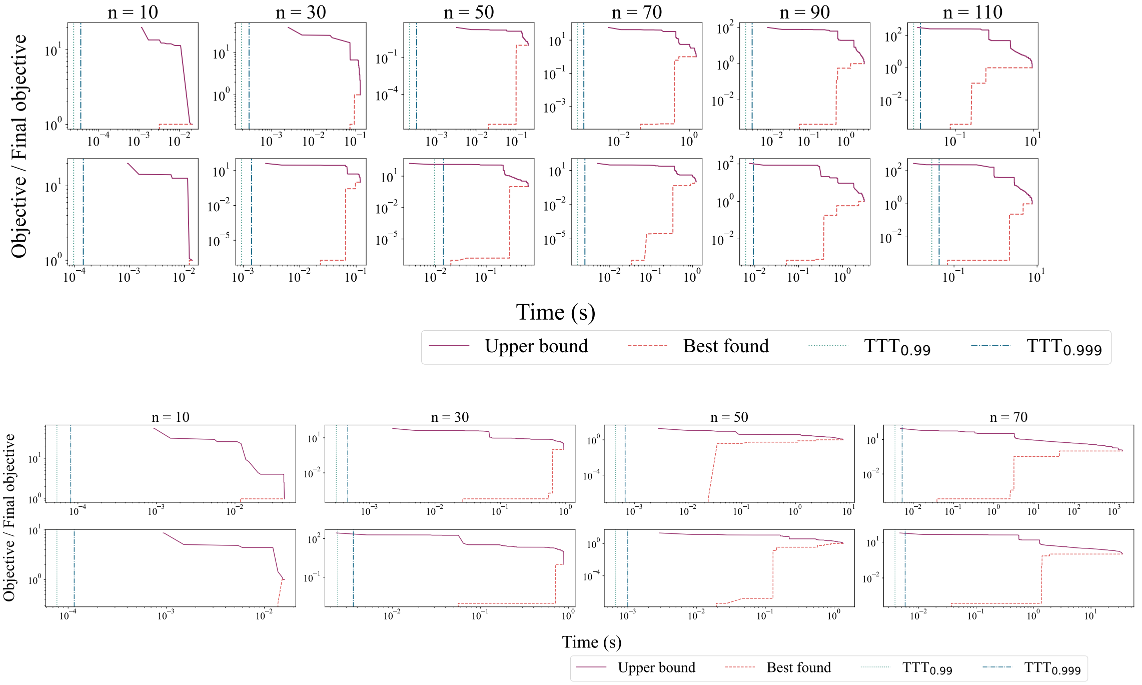


Fig. 11: This figure depicts sample trajectories of `Gurobi`'s upper and lower bounds against $\mathtt{TTT}_{0.99}$ and $\mathtt{TTT}_{0.999}$ for edge density $p = 0.5$ (above) and $p = 0.75$ (below). For each graph size, the top row represents the instance where the ratio between `Gurobi`'s solution time and $\mathtt{TTT}_{0.99}$ is the greatest, and the bottom row represents the instance where the ratio is the smallest–all instances were run with 100 sweeps. In most instances, `Neal` reaches the $\mathtt{TTT}_{0.999}$ confidence before `Gurobi` even returns a callback.

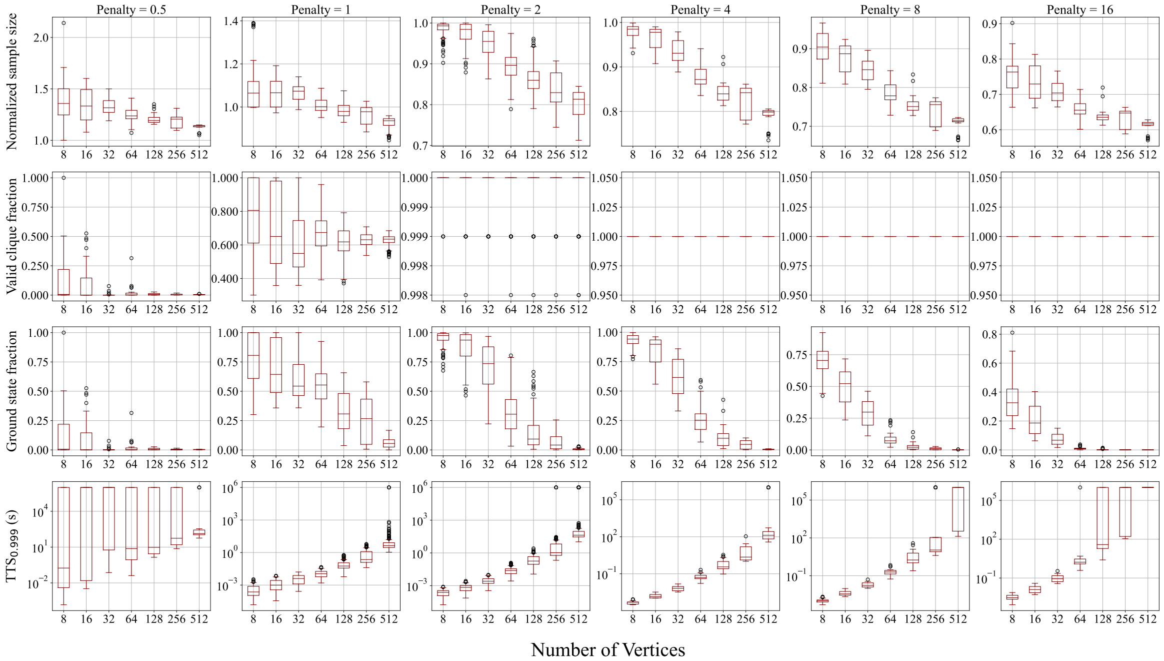


Fig. 12: This figure plots the normalized sample size (the size of the returned solution divided by the ground truth maximum clique size) and the fraction of reads that resulted in a valid clique for graph density $p = 0.5$. These figures were used to compute the fraction of reads resulting in a ground state solution and the corresponding $\mathtt{TTT}_{0.999}$ (also plotted). As the penalty weight is increased, the normalized sample size decreases, and the fraction of valid cliques increases. This highlights the delicate trade-off between constraints and the objective in penalty formulations.

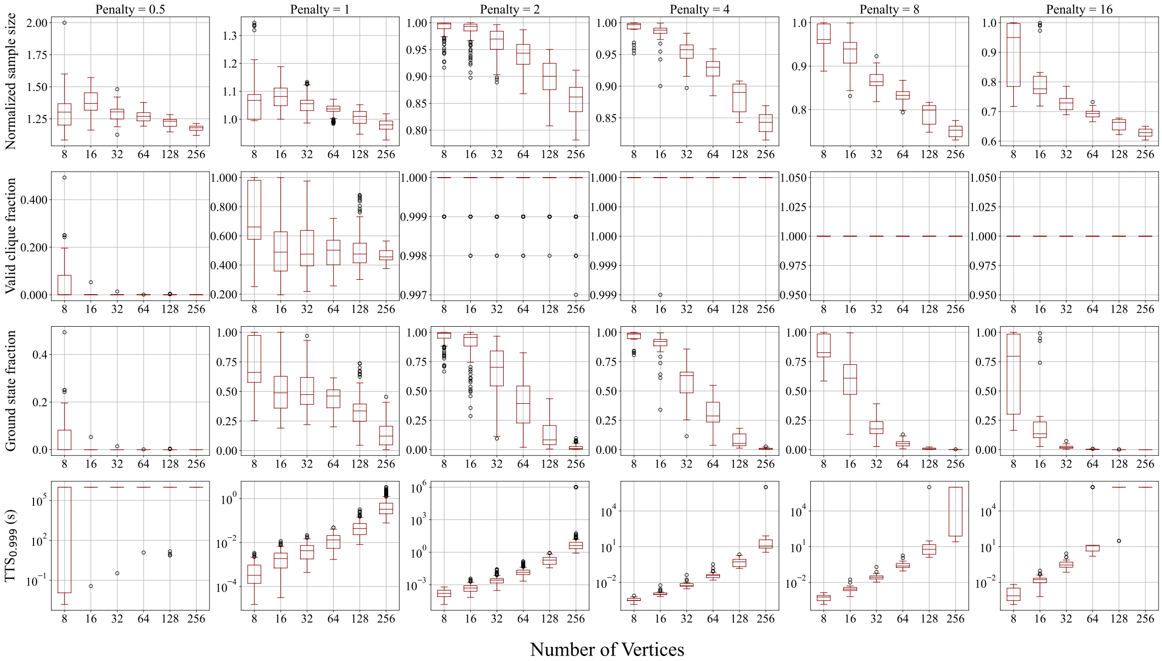


Fig. 13: This figure plots the normalized sample size (the size of the returned solution divided by the ground truth maximum clique size) and the fraction of reads that resulted in a valid clique for graph density $p = 0.75$. These figures were used to compute the fraction of reads resulting in a ground state solution and the corresponding $\texttt{TTT}_{0.999}$ (also plotted). As the penalty weight is increased, the normalized sample size decreases, and the fraction of valid cliques increases. This highlights the delicate trade-off between constraints and the objective in penalty formulations.