

Hybrid Quantum Branch-and-Bound Method for Quadratic Unconstrained Binary Optimization

Zedong Peng
Purdue University
610 Purdue Mall
47907, West Lafayette,
Indiana, USA
peng372@purdue.edu

Daniel de Roux
Carnegie Mellon
University*
5000 Forbes Ave
15213, Pittsburgh,
Pennsylvania, USA
dderoux@andrew.cmu.edu
*Daniel is now at Google.

David E. Bernal Neira
Purdue University
610 Purdue Mall
47907, West Lafayette,
Indiana, USA
dbernaln@purdue.edu

ABSTRACT

Quantum algorithms have shown promise in solving Quadratic Unconstrained Binary Optimization (QUBO) problems, benefiting from their connection to the transverse field Ising model. Various Ising solvers, both classical and quantum, have emerged to tackle such problems efficiently but lack global optimality guarantees and oftentimes suffer from hardware limitations such as limited qubit availability. In this work, we propose a hybrid branch-and-bound (B&B) framework that integrates Ising solvers as heuristics within a classical B&B algorithm. Unlike prior theoretical studies, our work presents a practical implementation, available as open-source at GitHub. We explore when and where to apply Ising solvers in the search tree, and introduce a custom branching rule optimized QUBO embedding. Our method is evaluated on hundreds of QUBO instances from QUBOLib.jl using Gurobi and the D-Wave quantum annealer. Our Results show up to 11% less solution time and 17% fewer nodes compared to default Gurobi, an off-the-shelf commercial optimization solver. These findings demonstrate the value of hybrid quantum-classical strategies for enhancing exact optimization.

Keywords

Quadratic Unconstrained Binary Optimization, Branch-and-Cut, Quantum Optimization.

1 INTRODUCTION

The quadratic unconstrained binary optimization problem (QUBO) is a central family of optimization programs where one seeks to minimize a quadratic function over binary variables. Formally, a QUBO is defined as

$$\begin{aligned} &\text{Minimize} \quad \mathbf{x}^\top \mathbf{Q} \mathbf{x} \\ &\text{subject to} \quad x_i \in \{0, 1\}, \quad \forall i \in N, \end{aligned} \quad (1)$$

where $\mathbf{x} \in \{0, 1\}^n$, $N = \{1, \dots, n\}$, and $\mathbf{Q} \in \mathbb{R}^{n \times n}$ is symmetric. The objective function includes both linear terms $Q_{ii}x_i$ and bilinear terms $Q_{ij}x_i x_j$.

This family of optimization problems arises in several applications such as MAXCUT, graph partitioning, protein folding, machine learning pipelines and many others [GKHD22]. QUBO is a special case of mixed-integer programming (MIP) with only binary variables, no explicit constraints, and quadratic objective, which in turn is a subfamily of Mixed-integer nonlinear programming (MINLP), extending MIP to allow general nonlinear objective functions and constraints. By this token, QUBOs can enjoy the rich set of methods and solver capabilities that have been steadily expanding over the past three decades

—from mixed-integer linear programs (MILPs), to second-order cone programs (MISOCPs), to nonconvex quadratic problems (MIQP/MIQCP), and finally to general MINLPs [Bix12, KBPV22, Gur24]— to solve these problems to global optimality.

MINLPs are, in general, hard to solve as they often involve nonconvexities such as bilinear terms, nonlinear constraints, or logic-based disjunctions. These break convexity and often require reformulations, decompositions, or restrictions to tractable subclasses [KBLG19]. Most approaches to solve MINLPs to global optimality rely on the branch-and-bound (B&B) algorithm, enhanced with methods like McCormick relaxations, Reformulation-Linearization Technique (RLT) cuts, and second-order cone approximations [RKS23]. Modern commercial solvers such as Gurobi incorporate these techniques and can solve QUBOs exactly to optimality. However, exact methods quickly become intractable as problem size grows.

To handle such large instances, heuristic solvers are commonly used. Metaheuristics such as simulated annealing, tabu search, genetic algorithms, large-neighborhood search, and path-relinking can produce high-quality solutions for problems with tens or

hundreds of thousands of variables. More recently, physics-inspired methods such as coherent Ising machines, digital annealers, and quantum annealers have shown promise in efficiently sampling low-energy states [MMB22]. While these methods do not guarantee global optimality, they offer practical performance on large-scale QUBO instances.

This work explores hybrid approaches that pair these solvers with quantum and physics-inspired heuristics to accelerate B&B while preserving global optimality guarantees. That is, providing both a solution and a certificate of global optimality.

Motivation for Hybrid Algorithms

While classical exact methods such as branch-and-bound (B&B) can solve small to medium-sized QUBOs to global optimality, they become computationally impractical for large problems due to the exponential growth in subproblems and the cost of solving relaxations. This motivates using quantum heuristics, thanks to the connection between QUBOs and the transverse field Ising mode [Luc14]. However, current quantum hardware is limited by qubit count, sparse connectivity, and environmental noise [AAA⁺24]. In fact, practical instances typically exceed hardware capacity which often times prevents direct execution, or obtained solutions are of poor quality. More importantly, global optimality usually cannot be guaranteed by quantum hardware.

These complementary limitations motivate hybrid strategies that combine classical guarantees with quantum or physics-inspired heuristics. Quantum heuristics can guide branching, supply high-quality incumbents, or tighten bounds, while the classical solver retains responsibility for global optimality certificates. The B&B tree structure naturally supports this integration by enabling selective oracle calls on promising subproblems.

Our Contribution

We present a practical hybrid quantum-classical algorithm for solving QUBO problems by embedding Ising-based heuristics into a classical B&B framework. Our method interfaces externally with `Gurobi` and invokes classical or quantum Ising solvers as heuristic oracles on selected subproblems. These solvers are invoked at different levels of the B&B tree. Oracle calls may provide branching decisions or candidate incumbent solutions. To improve effectiveness, we apply a preprocessing step to reduce the size of embedded subproblems and use a graph-aware branching rule based on variable degrees in the QUBO interaction graph. This design maximizes the value of Ising oracle calls while remaining within hardware constraints. Our implementation integrates with both simulated and hardware Ising

solvers and is publicly released¹, together with a benchmark suite of over five thousand QUBO instances from `QUBOLib.jl`².

- **Algorithmic contributions:** (i) a decision mechanism that triggers an Ising oracle at different stages of the branch and bound process; (ii) a graph-aware branching rule coupled with edge-contraction preprocessing.
- **Empirical contributions:** an open-source implementation and benchmark over more than a thousand instances show a median 17% node reduction and 11% wall-clock speed-up. The shifted geometric mean with a 10-second shift (SGM10) reduces baseline solve time from 154 s to 137 s.

The remainder of this paper is organized as follows. Section 2 reviews exact, heuristic, and quantum approaches. Section 3 describes the hybrid algorithm. Section 4 presents empirical results.

Related Work

Quantum-centric hybrid methods.

Theoretical groundwork for integrating quantum routines into classical branch-and-bound (B&B) was laid in [Mon20], which formalized the *Quantum Branch and Bound* (QBB) model and proved potential speedups when quantum subroutines tighten bounds or guide branching. Further theoretical developments appear in [CMYP22]. Building on these ideas, several prototype implementations have been proposed. The work in [STE24] inserts D-Wave quantum-annealing calls at selected B&B nodes to refine incumbents on small QUBO benchmarks. The method in [MHNY24] proposes Quantum Relaxation-Based B&B (QR-BnB), where a gate-model device estimates ground-state energies to tighten lower bounds. Reference [SRC⁺25] introduces Branch-and-Bound Digitized Counterdiabatic Quantum Optimization (BB-DCQO), which branches on spins with high measurement uncertainty to focus the search. The architecture in [HBBZ24] sketches a forward-compatible design to offload entire subproblem relaxations to future quantum devices. Collectively, these works demonstrate two main QBB insertion points, *bounding* and *branching*, while differing in hardware platform, problem size, and optimality guarantees.

¹ <https://github.com/SECQUOIA/QuantumBranchAndBound>

² <https://github.com/SECQUOIA/QUBOLib.jl>

Classical-centric enhancements.

Classical improvements to B&B focus on structural preprocessing, parameter tuning, and exploiting problem structure. The study in [RKS23] shows that exploiting sparsity, reducing symmetry, and tuning parameters in the open-source MIP solver SCIP can significantly accelerate MaxCut (a special case of QUBO) on large, sparse graphs.

Our approach.

We target currently available hardware and avoid modifying solver internals. Rather than embedding quantum logic into the solver, we treat classical and quantum Ising engines as *external oracles* invoked through callbacks. These oracles provide candidate incumbents and branching guidance, while the commercial solver remains responsible for bounding, node selection, and global optimality certificates. This black-box strategy differs from QBB approaches that hard-code quantum subroutines or require customized solver forks. It also enables a fair, large-scale empirical study across thousands of diverse QUBO instances.

2 BACKGROUND

This section provides concise background on the three methodological pillars relevant to hybrid optimization: exact B&B, classical heuristics, and quantum and physics-inspired methods.

2.1 Exact Methods: Branch-and-Bound and Variants

The branch-and-bound (B&B) method is a recursive divide-and-conquer algorithm for solving mixed-integer optimization problems (MIPs) to global optimality. It systematically maintains upper and lower bounds on the optimal value and prunes regions of the search space that cannot contain better solutions. As a result, B&B is an exact method that can find and certify optimal solutions under general assumptions. We refer to [CCZ⁺14] for mixed-integer linear programming and to [BL12] for nonlinear generalizations. The typical steps of the B&B algorithm are:

1. **Initialization (Root Node):** Solve the continuous relaxation of the original MIP, ignoring integrality constraints. This yields an initial bound on the optimal objective value. If the solution is already integer feasible, it is globally optimal.
2. **Branching:** If the relaxation has fractional values for integer variables, choose one such variable and create subproblems by imposing disjunctions (e.g., $x_i \leq \lfloor x_i^* \rfloor$ or $x_i \geq \lceil x_i^* \rceil$), thereby partitioning the feasible region.
3. **Bounding:** Solve the continuous relaxation for each subproblem. The solution provides a bound (upper or lower, depending on the problem direction) on the objective within that subregion.
4. **Fathoming (Pruning):** Discard a node if: (i) its relaxation is infeasible, (ii) its bound is worse than the incumbent, or (iii) its relaxed solution is integer feasible. If the new feasible solution improves the incumbent, update the incumbent.
5. **Node Selection:** Select the next active node for exploration. Strategies include best-bound, depth-first, or breadth-first. The goal is to explore promising parts of the tree while managing memory.
6. **Iteration:** Repeat branching, bounding, and fathoming on the selected node. This recursive process explores the full tree or continues until optimality is proven.
7. **Termination:** Stop when no active nodes remain. The incumbent is then the optimal solution. If no feasible solution was found, the problem is infeasible.

While the branch-and-bound framework applies to both linear and nonlinear MIPs, solving the relaxations at each node can be computationally intensive. These relaxations are typically continuous and convex. For MILPs, they are linear programs (LPs), but more general MIPs may involve convex nonlinear relaxations. In some cases, these relaxations may even be nonconvex. Subproblems created during branching can be nearly as difficult to solve as the original problem. This observation has motivated the use of heuristics to either bypass expensive relaxations or produce feasible solutions that accelerate the search. Three common B&B variants that exploit this idea are Branch-and-Prune, Branch-and-Cut, and Primal Heuristics [LB96].

Branch and Prune: This variant focuses on recursive branching and pruning of subregions that cannot yield better solutions than the incumbent. Pruning decisions rely on bounds computed from continuous relaxations. A node is discarded if its relaxation is infeasible, yields a bound worse than the incumbent, or results in an integer-feasible solution. Tighter bounds enable earlier pruning and improve convergence.

Branch and Cut: This method augments B&B by adding cutting planes to the relaxed subproblems. These valid inequalities are violated by the current relaxation but respected by all feasible integer solutions. Effective cuts, such as Gomory, MIR, or cover inequalities, can tighten relaxations and improve pruning. Relaxations at nodes are typically

continuous and convex. For MILPs, this means LPs; for more general MIPs, they may be convex nonlinear programs. Although cutting can improve performance, excessive or poorly selected cuts may slow down per-node solution times, so cut management must be judicious.

Primal Heuristics: These methods generate high-quality feasible solutions early in the search. Stronger incumbent solutions improve pruning efficiency by tightening the upper bound. Common heuristic strategies include rounding, diving, local search, and large neighborhood search [Ber06].

- **Rounding:** Converts solutions from a relaxation into integer-feasible ones using rounding rules, followed by feasibility repair if needed.
- **Diving:** Fixes variables based on their fractional values and recursively solves subproblems to explore promising regions.
- **Local Search:** Improves a given solution by exploring its neighborhood using swap, flip, or exchange moves.
- **Large Neighborhood Search (LNS):** Relaxes a subset of variables and resolves the resulting reduced problem to escape local optima.

2.2 Heuristic Algorithms

Given the NP-hardness of QUBO, exact methods become impractical for large-scale instances. To address this, the literature has proposed a wide range of heuristics and approximation algorithms capable of producing high-quality solutions within reasonable computational time. In many cases, heuristics have been shown to recover optimal solutions on benchmark instances.

Simulated Annealing (SA) is a widely used metaheuristic inspired by the physical process of annealing in metallurgy. SA explores the solution space by iteratively proposing neighboring solutions [KGJV83]. Improvements are always accepted, while worse solutions may be accepted with a probability that decreases over time according to a predefined cooling schedule. This mechanism helps the algorithm escape local minima and explore diverse regions of the search space. The performance of SA is highly sensitive to parameter tuning, especially the cooling rate. For QUBO problems, several effective implementations have been reported in [AHA98, Bea98].

Local Search heuristics form another important category. In particular, *Tabu Search* uses a short-term memory structure (the tabu list) to prevent cycling and encourage diversification. At each iteration, the best admissible neighbor is selected, potentially even if it worsens the objective. This process allows escape from local optima. Detailed implementations can be found

in [Bea98, GKA98]. *Iterated Local Search* combines local search with strategic perturbations. Effective variants for QUBO are given in [Pal04].

Genetic Algorithms (GAs) represent a population-based heuristic that mimics evolutionary processes. A pool of candidate solutions evolves through crossover (recombination) and mutation (random perturbation). Selection is based on fitness, typically measured by the QUBO objective value. Many GA implementations incorporate local search to refine offspring, although convergence can be slow and quality may vary. These methods are computationally demanding and sensitive to population diversity. Relevant studies include [DSFC05, HAA00, MF99].

Additional heuristic families include *path relinking* [FPRR02], *cross-entropy methods* [LDM09], *global equilibrium search* [PPSS08], and *greedy construction strategies* [FPRR02]. For a comprehensive empirical comparison across heuristics for QUBO and Max-Cut, we refer to the systematic evaluation in [DGS18].

2.3 Quantum and Physics-Inspired hybrid Methods

Several quantum computing paradigms and their corresponding hardware implementations have been proposed to tackle difficult optimization problems. A prominent example is *Adiabatic Quantum Computing (AQC)*, which begins with a quantum system in the ground state of an initial Hamiltonian and slowly evolves it toward a cost Hamiltonian encoding the optimization objective. If the evolution is sufficiently slow, the adiabatic theorem suggests the system remains in the ground state of the final Hamiltonian, corresponding to the optimal solution. However, determining how slow is "sufficiently" slow depends on the minimum spectral gap during evolution, which is generally intractable to compute. As a result, real implementations use heuristics for schedule selection and face challenges such as thermal noise, decoherence, and limited qubit connectivity. These limitations often require embedding logical variables using multiple physical qubits, which adds overhead. *Quantum Annealing (QA)* models AQC in the presence of such physical imperfections and is used as a heuristic optimization method.

An alternative to AQC is the gate-based quantum computing model. In this framework, *Variational Quantum Algorithms (VQAs)* have emerged as a family of hybrid quantum-classical methods suitable for optimization [CAB⁺21]. VQAs rely on parameterized quantum circuits whose performance is evaluated by a classical optimizer based on a measured cost function. The classical optimizer updates the parameters to minimize this objective, typically through iterative

feedback. Training these circuits is known to be NP-hard in general [BK21].

Among VQAs, the *Quantum Approximate Optimization Algorithm (QAOA)* has become a widely studied strategy for combinatorial problems [WWJ⁺20]. QAOA alternates between applying the cost and mixing Hamiltonians. The number of alternations determines the circuit depth. Each round involves optimizing a set of continuous parameters, known as rotation angles, that control the unitary operations. Although QAOA is provably optimal in the infinite-depth limit due to its equivalence to AQC, its performance at finite depth remains difficult to analyze because of quantum many-body interactions and classical optimization difficulties [UB21].

We point out that all (VQAs) [CAB⁺21], and the Quantum Approximate Optimization Algorithm (QAOA) [FGG14], together with Ising-based hardware such as D-Wave quantum annealers [JAG⁺11], and coherent Ising machines [HSI⁺21] are specifically designed to tackle QUBOs heuristically. However, hardware noise and embedding overhead still limit scale, but empirical gains on medium-sized problems are encouraging [MMB22]. For a broader review of quantum heuristics for Ising problems, we refer the reader to [SBC⁺20].

As pointed out in the introduction, Quantum Branch-and-Bound (QBB) frameworks integrate such routines into classical B&B, aiming to accelerate bounding or branching while preserving global optimality guarantees [MHNY24, SRC⁺25, STE24].

3 PROPOSED METHODS

In this work, we propose and implement a hybrid quantum branch-and-bound (B&B) algorithm specifically designed for solving QUBO problems. The core idea is to incorporate heuristic solutions obtained from quantum hardware into the B&B tree to tighten the upper bound and enhance pruning efficiency. In general, modern B&B solvers allow external solution information to be injected in three ways: MIPStart (also known as warm start), heuristic callbacks, and variable hints. Since quantum solvers typically provide complete feasible solutions rather than partial guidance, our method focuses on the first two mechanisms and does not consider variable hints, which are better suited for soft guidance rather than hard feasible inputs.

Algorithm 1 shows the high-level pseudocode of our method. Compared to the standard B&B algorithm, our approach introduces three key enhancements. First, quantum solutions are injected at the root node using the MIPStart mechanism, allowing the solver to begin with a high-quality incumbent and prune large portions of the tree early on. Second, we extend this injection strategy to subtrees by invoking heuristic callbacks at

interior nodes. This enables the algorithm to continually benefit from quantum-generated solutions throughout the search. Even though hybrid quantum solvers can handle QUBO problems beyond the size limits of quantum annealers, the quality and efficiency of quantum solutions tend to degrade with problem size. To better exploit quantum hardware, we design branching strategies that prioritize subproblems likely to be smaller and thus more amenable to high-quality quantum solutions. These methods are implemented in our library and extensively tested on thousands of QUBO instances. More details about the experimental results are presented in the next section.

Algorithm 1 Quantum Branch-and-Bound Framework

```

1: Inject solution ▷ via MIPStart
2: Calculate branch priority ▷ based on Q matrix
3: Perform presolve
4: Solve root node LP relaxation
5: while termination criteria not met do
6:   Node selection
7:   Inject solution ▷ via heuristic callback
8:   Node presolve
9:   Solve the LP relaxation
10:  Apply cutting planes
11:  Apply primal heuristics
12:  if a feasible integer solution is found then
13:    Update incumbent solution
14:  else if the node is still feasible then
15:    Branch on fractional variables
16:    Insert child nodes into the search tree
17:  end if
18: end while

```

3.1 Root node: MIPstart

Quantum solvers, such as quantum annealers, are designed to solve QUBO problems by reformulating them as equivalent Ising models that can be directly mapped onto quantum hardware. For large-scale QUBO instances that exceed the capacity of physical quantum annealers, hybrid solvers combine classical and quantum resources to solve the full problem without manual decomposition. This makes it possible to directly apply quantum solvers to the original QUBO problem. Although such solvers can return feasible solutions, their quality is not always guaranteed, especially for problems with many variables or complex landscapes [PND08]. Nevertheless, these solutions can still provide useful upper bounds for minimization problems, and can be injected into the root node of a branch-and-bound solver using *MIPStart*, which allows users to supply one or multiple feasible solutions to guide the search. This process corresponds to Step1 in Algorithm 1.

3.2 Injection at internal nodes

An extension of the root-node injection idea is to insert solutions at internal nodes, finding a high-quality solution to each branch of the tree. These insertions can be handled by *heuristic callbacks*, a technique available in modern MIP solvers that allows users to supply a feasible solution dynamically during the tree search. These callbacks are implemented in step 7 of Algorithm 1. However, these calls must be handled with care, as we now discuss. Invoking heuristic solvers at every node can be prohibitively expensive and may significantly increase the overall computation time. We consider the following strategy to mitigate this. Since QUBO does not have constraints, a feasible solution for one node in the tree is feasible for all its child nodes. This suggests that instead of obtaining heuristic solutions at each node, we rather find a large set of high-quality solutions *a priori*, i.e., before starting the branch and bound algorithm, and store these solutions for later use. This strategy also aligns well with the high-throughput nature of quantum solvers, which are capable of producing a large volume of feasible solutions.

3.3 Embedding and Branching Priority

To solve an arbitrarily posed binary quadratic problem directly on a D-Wave system requires mapping, called minor embedding, to the QPU Topology of the system's quantum processing unit (QPU) [OOTT19]. By default, DWave will call *minorminer* to find the embedding of the input QUBOs. Because the quantum processing unit is inherently limited by the number of qubits, it is desirable to embed problems with a smaller number of variables. Figure 1 shows an example of an embedding for a 3-variable QUBO (2) onto a four-node QPU topology [D-Wnd]. The QUBO problem is first represented by the triangular graph, where nodes represent variables and edges represent the quadratic terms. Embedding aims to map the triangular graph into the fully connected and sparse four-node graphs.

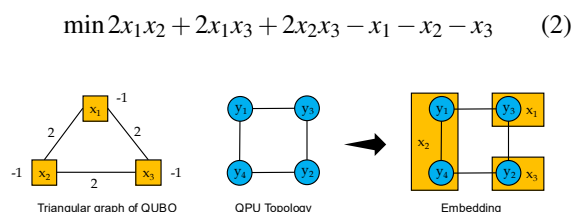


Figure 1: Example of embedding a 3-variable QUBO problem

This suggests that one should call a heuristic solver precisely on the nodes where most variables have been fixed, which are either deep nodes (i.e., further down the tree) or nodes where variables that appear in a large number of quadratic terms have been fixed. To illustrate, consider the following example. Suppose that we

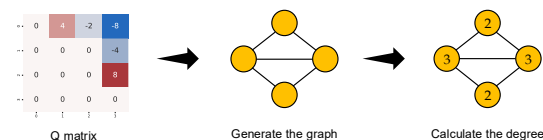


Figure 2: Example of calculating branch priority from quadratic objective matrix

are given a QUBO with Q matrix as in Figure 2, which gives the optimization problem

$$\min_{x \in \{0,1\}^4} 4x_1x_2 - 2x_1x_3 - 8x_1x_4 - 4x_2x_4 + 8x_3x_4 \quad (3)$$

Observe now that if x_1 and x_4 are fixed, we obtain an optimization program on variables x_2 and x_3 without quadratic terms. This toy example suggests that a good rule of thumb consists of first branching on variables that appear in many quadratic terms in the objective function, as they have a larger potential to diminish the size of the problem when branching on them.

Formally, we consider the matrix graph $G(Q)$ of Q where the vertices correspond to the variables in Program 1, and we add an edge between two vertices i and j if $Q_{ij} \neq 0$. The *degree* of a vertex $d(i)$ is the number of edges incident to it, or equivalently, the number of quadratic terms of the form $Q_{ij}x_ix_j, j \in N$ in which the variable x_i appears in the objective. We define the *Branch priority* of a vertex of $G(Q)$ as its degree. Figure 2 shows an example of computing branch priority for QUBO problem (3). In the branching step of our proposed B&B algorithm, we continue the iteration on the branch with the highest branch priority, breaking ties arbitrarily. This step is implemented in lines 2 and 15 of Algorithm 1.

4 NUMERICAL EXPERIMENTS

We evaluate the proposed Quantum Branch-and-Bound method on a benchmark set of 5807 instances from QUBOLib, which includes planted solutions to 3-regular 3-XORSAT and 5-regular 5-XORSAT problems. Figure 3 and Table 1 provide a statistical overview of these instances. Figure 3 illustrates the quadratic sparsity of the QUBOLib instances, revealing a clear trend: as the number of variables increases, the quadratic term sparsity also increases. Table 1 categorizes the full benchmark set into three collections. The 3-regular 3-XORSAT problems are sourced from two different arXiv datasets, while the 5-regular 5-XORSAT problems cover significantly larger problem sizes, with up to 24,576 variables.

Our Quantum Branch-and-Bound algorithm is implemented using a modular and extensible Julia-based pipeline. We begin by loading over 5000 QUBO

instances from the QUBOLib benchmark using the QUBOLib.jl and QUBOTools.jl [XRA⁺23] packages. Each instance is translated into a structured model using the JuMP modeling language.

To guide the branching decisions within the solver, we calculate the degree of the graph induced by the quadratic objective matrix using Graphs.jl. The branching priority information is passed to Gurobi 11.0.0, utilizing its Branch & Cut capabilities for exact optimization. The NonConvex parameter is set to 2 to enable solving non-convex quadratic programs, which are reformulated into bilinear forms and handled via spatial branching. All experiments are run with ThreadLimit = 1 and a time limit of 900 seconds.

In addition to exact methods, we incorporate heuristic warm-starts from quantum solvers such as D-Wave via MQLib [DGS18], allowing the solver to initialize with high-quality feasible solutions. We test 16 heuristic methods from MQLib, including BURER2002, FESTA2002GVNSPR, PALUBECKIS2004bMST3, PALUBECKIS2006, FESTA2002GPR, FESTA2002GVNS, MERZ2004, PALUBECKIS2004bMST2, BEASLEY1998TS, LU2010, FESTA2002G, PALUBECKIS2004bMST1, MERZ1999GLS, MERZ2002KOPT, ALKHAMIS1998, MERZ2002GREEDYKOPT. We also evaluate simulated annealing using dimod.neal (v0.5.9), and quantum annealing on D-Wave's Advantage 4.1 system with 5,750 qubits and over 35,000 couplers.

To focus our analysis, we filter the dataset to instances that (i) take more than 10 seconds to solve using default Gurobi, and (ii) can be solved to optimality by at least one of the tested methods within the time limit. This yields a refined test set of 1,454 instances for detailed comparison.

We measure performance using the shifted geometric mean (SGM) of solve time and number of explored nodes, with a shift of 10 (SGM10). If the instance is not solved to optimality within the time limit, the solve time is always set to the corresponding time limit and we record the number of explored nodes. The results are presented in Table 4.

Notably, MQLib only returns the best-found solution to the given problem. When used with Gurobi's MIP start strategy, this solution is injected at the root node. Alternatively, when used in a heuristic callback strategy, MQLib is invoked at every node to attempt solving subproblems. For SA and QA, we experiment with injecting the top 1, 10, 30, or 100 solutions sorted by objective value. In callback mode, SA and QA are applied once at the root node to generate a solution pool. During the branch-and-bound process, solutions from this pool are selectively injected based on the node subproblem. Moreover, to test the upper bound of the im-

provement, we tested the performance of providing the best solution in the MIP start strategy.

The results are summarized in Table 4. Among the 16 tested MQLib heuristics, PALUBECKIS2006 consistently achieves the best performance. To simplify the presentation, we only report the results of PALUBECKIS2006 in Table 4 as the representative MQLib method. It is shown that using branch priority alone improves Gurobi's performance by 17.3% in node count and 11.1% in runtime. When using MIP start with PALUBECKIS2006, simulated annealing, or quantum annealing, we observe approximately 10% runtime improvement. For simulated annealing, injecting more solutions leads to a modest 3% additional improvement, while quantum annealing shows limited sensitivity to the number of solutions provided. Combining MIP start with branch priority yields better results than MIP start alone, but still slightly underperforms the branch priority strategy alone. The results of injecting the best solution demonstrate the upper bound of improvement achievable via solution injection: the runtime is reduced by 83.0%, and the number of explored nodes drops by 90.6%, while still solving 1170 out of 1454 instances. Even though the results remains similar when combined with priority, it confirms the potential of how much high-quality starts can dramatically accelerate solving.

When applying heuristic callbacks, we find that invoking MQLib at every node introduces significant overhead, leading to longer runtimes and a reduced solve rate of 921 out of 1,454 instances. While SA and QA callbacks are applied in a more efficient way and invoked only once and used via a solution pool, they still result in longer solve times and slightly fewer explored nodes. These findings suggest that node-level heuristic injection is often too costly in practice and should be used cautiously.

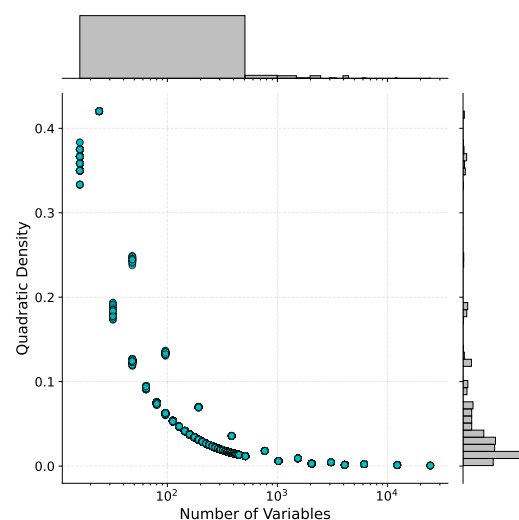


Figure 3: Quadratic sparsity of QUBOLib instances

Collection	# of instances	# of variables
3-Regular 3-XORSAT [KAHL22]	2300	16 ~ 4096
3-Regular 3-XORSAT [Hen19]	3200	16 ~ 4096
5-Regular 5-XORSAT [Hen19]	307	24 ~ 24576

Table 1: XORSAT Planted Solutions Collections - QUBOLib

5 CONCLUSIONS

This work proposes a practical hybrid quantum-classical branch-and-bound framework for solving QUBO problems to global optimality. The proposed method provides a unified framework to integrate Ising solvers, including both classical heuristics and quantum annealers, into a Gurobi-based branch-and-bound solver. Extensive experiments on over 5,800 instances from QUBOLib show that warm-starting with high-quality solutions from Ising solvers yields a 5% improvement and a carefully designed branch priority rule alone can reduce solve time and node count by over 10%. However, the improvement remains well below the potential upper bound obtained by providing the best solution. Additionally, node-wise heuristic callbacks are computationally expensive and often counterproductive. Overall, our results validate the potential of hybrid quantum-classical strategies to accelerate exact solvers on structured QUBO problems. Developing more effective methods for integrating quantum solvers as node-wise heuristics remains an open direction for future research.

6 ACKNOWLEDGMENTS

D.B.N. and Z.P. acknowledge the support of the startup grant of the Davidson School of Chemical Engineering at Purdue University.

7 REFERENCES

- [AAA⁺24] Amira Abbas, Andris Ambainis, Brandon Augustino, Andreas Bärtzsch, Harry Buhrman, Carleton Coffrin, Giorgio Cor-tiana, Vedran Dunjko, Daniel J Egger, Bruce G Elmegeen, et al. Challenges and opportunities in quantum optimization. *Nature Reviews Physics*, pages 1–18, 2024.
- [AHA98] Talal M Alkhamis, Merza Hasan, and Mo-hamed A Ahmed. Simulated annealing for the unconstrained quadratic pseudo-Boolean function. *European journal of op-erational research*, 108(3):641–652, 1998.
- [Bea98] John E Beasley. Heuristic algorithms for the unconstrained binary quadratic pro-gramming problem, 1998.
- [Ber06] Timo Berthold. *Primal heuristics for mixed integer programs*. PhD thesis, Zuse Institute Berlin (ZIB), 2006.
- [Bix12] Robert E Bixby. A brief history of linear and mixed-integer programming computa-tion. *Documenta Mathematica*, 2012:107–121, 2012.
- [BK21] Lennart Bittel and Martin Kliesch. Training variational quantum algorithms is NP-hard. *Physical review letters*, 127(12):120502, 2021.
- [BL12] Samuel Burer and Adam N Letchford. Non-convex mixed-integer nonlinear pro-gramming: A survey. *Surveys in Opera-tions Research and Management Science*, 17(2):97–106, 2012.
- [CAB⁺21] Marco Cerezo, Andrew Arrasmith, Ryan Babbush, Simon C Benjamin, Suguru Endo, Keisuke Fujii, Jarrod R McClean, Kosuke Mitarai, Xiao Yuan, Lukasz Cin-cio, et al. Variational quantum algorithms. *Nature Reviews Physics*, 3(9):625–644, 2021.
- [CCZ⁺14] Michele Conforti, Gérard Cornuéjols, Gia-como Zambelli, Michele Conforti, Gérard Cornuéjols, and Giacomo Zambelli. *Inte-ger programming models*. Springer, 2014.
- [CMYP22] Shouvanik Chakrabarti, Pierre Minssen, Romina Yalovetzky, and Marco Pis-toia. Universal quantum speedup for branch-and-bound, branch-and-cut, and tree-search algorithms. *arXiv preprint arXiv:2210.03210*, 2022.
- [D-Wnd] D-Wave Systems Inc. Example: Solv-ing a constrained sat problem. https://docs.dwavequantum.com/en/latest/quantum_research/example_sat_constrained.html, n.d. Accessed: 2025-05-30.
- [DGS18] Iain Dunning, Swati Gupta, and John Sil-berholz. What works best when? A sys-tematic evaluation of heuristics for Max-Cut and QUBO. *INFORMS Journal on Computing*, 30(3):608–624, 2018.
- [DSFC05] Abraham Duarte, Ángel Sánchez, Felipe Fernández, and Raúl Cabido. A low-level hybridization between memetic algorithm and VNS for the max-cut problem. In *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, pages 999–1006, 2005.
- [FGG14] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.
- [FPRR02] Paola Festa, Panos M Pardalos, Mauri-cio GC Resende, and Celso C Ribeiro.

Strategy	Heuristic	# solved instances	Node Count	Runtime [s]
Baseline	-	1065	28170.0	154.5
Branch Priority	-	1077	23304.8 (-17.3%)	137.4 (-11.1%)
MIP Start	MQLib PALUBECKIS2006	1073	24543.0 (-12.9%)	142.5 (-7.8%)
MIP Start	DWave SA TOP1	1070	25418.8 (-9.8%)	150.1 (-2.9%)
MIP Start	DWave SA TOP10	1080	25256.3 (-10.3%)	148.3 (-4.1%)
MIP Start	DWave SA TOP30	1074	24616.1 (-12.6%)	144.8 (-6.3%)
MIP Start	DWave SA TOP100	1059	24665.1 (-12.4%)	145.1 (-6.1%)
MIP Start	DWave QA TOP1	1081	24958.7 (-11.4%)	148.8 (-3.7%)
MIP Start	DWave QA TOP10	1074	24880.3 (-11.7%)	149.8 (-3.1%)
MIP Start	DWave QA TOP30	1069	25069.6 (-11.0%)	150.1 (-2.8%)
MIP Start	DWave QA TOP100	1079	25189.0 (-10.6%)	150.6 (-2.6%)
MIP Start + Embedding	DWave QA TOP1	1073	25713.4 (-8.7%)	148.2 (-4.1%)
MIP Start + Embedding	DWave QA TOP10	1078	24742.1 (-12.2%)	142.9 (-7.5%)
MIP Start + Embedding	DWave QA TOP30	1077	25093.5 (-10.9%)	146.0 (-5.5%)
MIP Start + Embedding	DWave QA TOP100	1069	25536.7 (-9.3%)	146.6 (-5.1%)
MIP Start	Best Solution	1170	2643.3 (-90.6%)	26.3 (-83.0%)
MIP Start + Branch Priority	MQLib PALUBECKIS2006	1094	23292.9 (-17.3%)	137.7 (-10.9%)
MIP Start + Branch Priority	DWave SA TOP1	1068	23440.7 (-16.8%)	138.0 (-10.7%)
MIP Start + Branch Priority	DWave SA TOP10	1079	23956.5 (-15.0%)	140.3 (-9.2%)
MIP Start + Branch Priority	DWave SA TOP30	1083	24000.7 (-14.8%)	140.5 (-9.1%)
MIP Start + Branch Priority	DWave SA TOP100	1077	23550.5 (-16.4%)	138.1 (-10.6%)
MIP Start + Branch Priority	DWave QA TOP1	1077	24312.4 (-13.7%)	142.3 (-7.9%)
MIP Start + Branch Priority	DWave QA TOP10	1072	23628.1 (-16.1%)	139.0 (-10.0%)
MIP Start + Branch Priority	DWave QA TOP30	1093	23714.2 (-15.8%)	138.9 (-10.1%)
MIP Start + Branch Priority	DWave QA TOP100	1071	24342.5 (-13.6%)	141.4 (-8.5%)
MIP Start + Branch Priority	Best Solution	1164	2649.6 (-90.6%)	26.5 (-82.9%)
Heuristic Callback	MQLib PALUBECKIS2006	921	17698.5 (-37.2%)	307.8 (+99.2%)
Heuristic Callback	DWave QA	1060	27785.0 (-1.4%)	160.5 (+3.9%)
Heuristic Callback	DWave SA	1039	26559.0 (-5.7%)	183.9 (+19.0%)

Table 2: Summary of solver performance on 1454 instances in QUBOLib (SGM10)

- Randomized heuristics for the MAX-CUT problem. *Optimization methods and software*, 17(6):1033–1058, 2002.
- [GKA98] Fred Glover, Gary A Kochenberger, and Bahram Alidaee. Adaptive memory tabu search for binary quadratic programs. *Management Science*, 44(3):336–345, 1998.
- [GKHD22] Fred Glover, Gary Kochenberger, Rick Hennig, and Yu Du. Quantum bridge analytics I: a tutorial on formulating and using QUBO models. *Annals of Operations Research*, 314(1):141–183, 2022.
- [Gur24] Gurobi Optimization, LLC. Gurobi Optimizer Reference Manual, 2024.
- [HAA00] Merza Hasan, Talal Alkhamis, and Jafar Ali. A comparison between simulated annealing, genetic algorithm and tabu search methods for the unconstrained quadratic Pseudo-Boolean function. *Computers & industrial engineering*, 38(3):323–340, 2000.
- [HBBZ24] Thomas Häner, Kyle EC Booth, Sima E Borujeni, and Elton Yechao Zhu. Solving QUBOs with a quantum-amenable branch and bound method. *arXiv preprint arXiv:2407.20185*, 2024.
- [Hen19] Itay Hen. Equation planting: a tool for benchmarking ising machines. *Physical Review Applied*, 12(1):011003, 2019.
- [HSI⁺21] Toshimori Honjo, Tomohiro Sonobe, Kensuke Inaba, Takahiro Inagaki, Takuya Ikuta, Yasuhiro Yamada, Takushi Kazama, Koji Enbutsu, Takeshi Umeki, Ryoichi Kasahara, et al. 100,000-spin coherent Ising machine. *Science advances*, 7(40):eabh0952, 2021.
- [JAG⁺11] Mark W Johnson, Mohammad HS Amin, Suzanne Gildert, Trevor Lanting, Firas Hamze, Neil Dickson, Richard Harris, Andrew J Berkley, Jan Johansson, Paul Bunyk, et al. Quantum annealing with manufactured spins. *Nature*, 473(7346):194–198, 2011.
- [KAHL22] Matthew Kowalsky, Tameem Albash, Itay Hen, and Daniel A Lidar. 3-regular three-xorsat planted solutions benchmark of classical and quantum heuristic optimiz-

- ers. *Quantum Science and Technology*, 7(2):025008, 2022.
- [KBLG19] Jan Kronqvist, David E Bernal, Andreas Lundell, and Ignacio E Grossmann. A review and comparison of solvers for convex MINLP. *Optimization and Engineering*, 20:397–455, 2019.
- [KBPV22] Thorsten Koch, Timo Berthold, Jaap Pedersen, and Charlie Vanaret. Progress in mathematical programming solvers from 2001 to 2020. *EURO Journal on Computational Optimization*, 10:100031, 2022.
- [KGJV83] Scott Kirkpatrick, C Daniel Gelatt Jr, and Mario P Vecchi. Optimization by simulated annealing. *science*, 220(4598):671–680, 1983.
- [LB96] Abilio Lucena and John E Beasley. Branch and cut algorithms. *Advances in linear and integer programming*, 4:187–221, 1996.
- [LDM09] Manuel Laguna, Abraham Duarte, and Rafael Marti. Hybridizing the cross-entropy method: An application to the max-cut problem. *Computers & Operations Research*, 36(2):487–498, 2009.
- [Luc14] Andrew Lucas. Ising formulations of many NP problems. *Frontiers in physics*, 2:74887, 2014.
- [MF99] Peter Merz and Bernd Freisleben. Genetic algorithms for binary quadratic programming. In *Proceedings of the genetic and evolutionary computation conference*, volume 1, pages 417–424. Morgan Kaufmann Orlando, FL, 1999.
- [MHNY24] Hiromichi Matsuyama, Wei-hao Huang, Kohji Nishimura, and Yu Yamashiro. Efficient Internal Strategies in Quantum Relaxation based Branch-and-Bound. *arXiv preprint arXiv:2405.00935*, 2024.
- [MMB22] Naeimeh Mohseni, Peter L McMahon, and Tim Byrnes. Ising machines as hardware solvers of combinatorial optimization problems. *Nature Reviews Physics*, 4(6):363–379, 2022.
- [Mon20] Ashley Montanaro. Quantum speedup of branch-and-bound algorithms. *Physical Review Research*, 2(1):013056, 2020.
- [OOTT19] Shuntaro Okada, Masayuki Ohzeki, Masayoshi Terabe, and Shinichiro Taguchi. Improving solutions by embedding larger subproblems in a D-Wave quantum annealer. *Scientific reports*, 9(1):2098, 2019.
- [Pal04] Gintaras Palubeckis. Multistart tabu search strategies for the unconstrained binary quadratic optimization problem. *Annals of Operations Research*, 131:259–282, 2004.
- [PND08] L Pusey-Nazzaro and P Date. Adiabatic quantum optimization fails to solve the knapsack problem. *arXiv 2020. arXiv preprint arXiv:2008.07456*, 2008.
- [PPSS08] Panos M Pardalos, Oleg A Prokopyev, Oleg V Shylo, and Vladimir P Shylo. Global equilibrium search applied to the unconstrained binary quadratic optimization problem. *Optimisation Methods and Software*, 23(1):129–140, 2008.
- [RKS23] Daniel Rehfeldt, Thorsten Koch, and Yuji Shinano. Faster exact solution of sparse MaxCut and QUBO problems. *Mathematical Programming Computation*, 15(3):445–470, 2023.
- [SBC⁺20] Yuval R Sanders, Dominic W Berry, Pedro CS Costa, Louis W Tessler, Nathan Wiebe, Craig Gidney, Hartmut Neven, and Ryan Babbush. Compilation of fault-tolerant quantum heuristics for combinatorial optimization. *PRX quantum*, 1(2):020312, 2020.
- [SRC⁺25] Anton Simen, Sebastián V Romero, Alejandro Gomez Cadavid, Enrique Solano, and Narendra N Hegade. Branch-and-bound digitized counterdiabatic quantum optimization. *arXiv preprint arXiv:2504.15367*, 2025.
- [STE24] Claudio Sanavio, Edoardo Tignone, and Elisa Ercolessi. Hybrid classical–quantum branch-and-bound algorithm for solving integer linear problems. *Entropy*, 26(4):345, 2024.
- [UB21] AV Uvarov and Jacob D Biamonte. On barren plateaus and cost function locality in variational quantum algorithms. *Journal of Physics A: Mathematical and Theoretical*, 54(24):245301, 2021.
- [WWJ⁺20] Madita Willsch, Dennis Willsch, Fengping Jin, Hans De Raedt, and Kristel Michielsen. Benchmarking the quantum approximate optimization algorithm. *Quantum Information Processing*, 19:1–24, 2020.
- [XRA⁺23] Pedro Maciel Xavier, Pedro Ripper, Tiago Andrade, Joaquim Dias Garcia, Nelson Maculan, and David E Bernal Neira. Qubo. jl: A julia ecosystem for quadratic unconstrained binary optimization. *arXiv preprint arXiv:2307.02577*, 2023.