LLM-QUBO: An End-to-End Framework for Automated QUBO Transformation from Natural Language Problem Descriptions

Huixiang Zhang[1], **Mahzabeen Emu**[2, 3], **Salimur Choudhury**[4]
[1]Department of Computer Science, Lakehead University, Thunder Bay, Ontario, Canada
[2]Quantum Communications and Computing Research Center, Memorial University of Newfoundland, St. John's, NL, Canada
[3]Department of Electrical and Computer Engineering, Memorial University of Newfoundland, St. John's, NL, Canada
[4]School of Computing, Queen's University, Kingston, Ontario, Canada
hzhan102@lakeheadu.ca, memu@mun.ca, s.choudhury@queensu.ca

Abstract

Quantum annealing offers a promising paradigm for solving NP-hard combinatorial optimization problems, but its practical application is severely hindered by two challenges: the complex, manual process of translating problem descriptions into the requisite Quadratic Unconstrained Binary Optimization (QUBO) format and the scalability limitations of current quantum hardware. To address these obstacles, we propose a novel end-to-end framework, LLM-QUBO, that automates this entire formulation-to-solution pipeline. Our system leverages a Large Language Model (LLM) to parse natural language, automatically generating a structured mathematical representation. To overcome hardware limitations, we integrate a hybrid quantum-classical Benders' decomposition method. This approach partitions the problem, compiling the combinatorial complex master problem into a compact QUBO format, while delegating linearly structured subproblems to classical solvers. The correctness of the generated QUBO and the scalability of the hybrid approach are validated using classical solvers, establishing a robust performance baseline and demonstrating the framework's readiness for quantum hardware. Our primary contribution is a synergistic computing paradigm that bridges classical AI and quantum computing, addressing key challenges in the practical application of optimization problem. This automated workflow significantly reduces the barrier to entry, providing a viable pathway to transform quantum devices into accessible accelerators for large-scale, real-world optimization challenges.

Introduction

Combinatorial optimization problems constitute a class of computational challenges of significant value in modern science, finance, logistics, and engineering. For canonical problems such as the Traveling Salesman Problem or portfolio optimization, the solution space grows exponentially with the problem size, rendering traditional computational methods intractable for large-scale instances. Quantum computing offers a promising alternative paradigm. Its core premise is that by encoding a problem's objective function as the energy function (Hamiltonian) of a physical system, a quantum annealer can naturally evolve to its lowest energy state, which corresponds to an optimal or near-optimal solution. The potential for quantum speedup is a primary driver for research, although current adiabatic quantum computing devices could not be proven to be faster than classical computing resources yet (Mücke, Gerlach, and Piatkowski 2023). This prospect has catalyzed extensive exploration in various applications, including financial asset allocation, biomedical research, and logistics network design (Morapakula et al. 2025; Oliveira, Silva, and Oliveira 2018; Malviya, Akash-Narayanan, and Seshadri 2023).

However, a stark reality on the path to quantum advantage is that we are in the era of noisy intermediate-scale quantum computing (Holliday 2025). Current quantum processors remain severely constrained in qubit count, coherence times, and connectivity (Morapakula et al. 2025). Consequently, the most viable path to realizing the potential of quantum computing in the foreseeable future lies not in purely quantum algorithms but in building Hybrid Quantum-Classical (HQC) systems (Zhao, Fan, and Han 2022). This paradigm advocates for a strategic division of labor: classical High-Performance Computing (HPC) systems handle tasks at which they excel, such as data pre- and post-processing, control flow, and numerically intensive computations, while quantum annealer function as specialized hardware accelerators or coprocessors, focused on solving the most computationally intensive and combinatorial complex parts of the problem. This synergistic model aims to merge the robustness of classical computation with the exploratory power of quantum computation, forming a powerful platform whose capabilities extend beyond those of any single paradigm alone (Zhao et al. 2025).

The central idea of this paper is that while the HQC architecture charts a path toward practical quantum computing, a fundamental obstacle, which we term the Formulation Bottleneck, blocks the realization of its potential. In the quantum annealing paradigm, any problem must be converted to the Quadratic Unconstrained Binary Optimization (QUBO) format as a middle layer between classical optimization problems and quantum hardware, minimizing an objective function of the form $y = x^T Q x$ (Glover et al. 2022). However, translating real-world problems into this representation is an exceptionally complex, expert-dependent, and error-prone task. It involves specialized skills such as constraint to penalty term translation, manual penalty coefficient tuning, and selecting appropriate QUBO precision, which not only raises the barrier to entry but also complicates integration

Copyright © 2025, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

into existing HPC workflows (Ayodele 2022; Volpe et al. 2024). The essence of this challenge is the lack of a high-level abstraction layer, analogous to the compilers in classical computing that automatically translate high-level code into machine instructions (Zaman, Tanahashi, and Tanaka 2022). To bridge this gap, our framework provides this necessary layer by creating an end-to-end pipeline that automates the transformation from a high-level problem description into an optimized QUBO matrix. This automated pipeline fundamentally transforms the skill set required to apply quantum computing. Our framework encapsulates the underlying complexity, shifting the primary challenge for a user from "How do I construct a valid QUBO?" to "How do I precisely formulate my optimization problem?" This skill is the root of classical operations research, not quantum computing. This reduction in the specialized knowledge barrier is fundamental to fostering a robust ecosystem of quantum optimization tools and accelerating their adoption across science and industry.

This paper presents a comprehensive framework that achieves this goal through the following key contributions.

- **An LLM-driven compiler for end-to-end QUBO formulation.** We propose and implement a novel framework that leverages an LLM to automate the transformation from a high-level problem description into a quantum-ready QUBO matrix, significantly lowering the barrier to entry for quantum optimization.
- **A validated HQC workflow for scalability.** We integrate Benders' Decomposition into our automated pipeline to tackle large-scale problems. This hybrid approach partitions a problem into a QUBO master problem suitable for quantum annealers, and subproblems solved by classical HPC resources, providing a viable pathway to solve problems that exceed the capacity of standalone quantum processors.
- **A novel integration of AI, HPC and quantum.** We demonstrate for the first time a seamless workflow that combines a generative AI compiler with a hybrid quantum decomposition solver, establishing a new paradigm for automated problem solving.

Related Work

We review related work by deconstructing the pipeline for transforming a combinatorial optimization problem into its QUBO representation. This pipeline includes several critical stages: problem formulation from natural language, binarization of variables, conversion of constraints into penalty terms, and the final generation of the QUBO matrix.

Optimization Problems Formulation From Natural Language Description

The task of converting a problem described in natural language into a formal mathematical model is a significant challenge that has traditionally required deep domain expertise. The emergence of LLM has catalyzed a field of research known as auto formulation, aimed at automating this process (Huang et al. 2025). Foundational efforts, such as the NL4Opt Competition, spurred the development of learning-based methods by structuring the task into subproblems like entity recognition and logical form generation (Ramamonjison et al. 2021, 2022). This work highlighted core challenges, including handling unstructured inputs and the need for models to generalize across different problem domains. Subsequent research has benchmarked prominent LLMs on this task, where large models such as GPT-4 established new performance levels without requiring the input of entities of prior baselines. The same work also introduced the LM4OPT framework, revealing a persistent capability gap between large models and smaller, fine-tuned ones when processing lengthy and complex problem contexts (Ahmed and Choudhury 2024). Other research has explored different frameworks and methodologies. For example, Jiang et al. introduced the LLMOPT framework, which employs a learning-based method to fine-tune LLMs to automate MILP formulation, showing a significant increase in the precision of the solution (Jiang et al. 2025). In a different paradigm, OptiChat was developed as an LLM-assisted interactive dialogue system. Instead of formulating problems from scratch, it helps practitioners interpret and query existing optimization models by augmenting the LLM with targeted code generation to ensure trustworthy responses (Chen et al. 2025).

QUBO Matrix Generation

The AutoQUBO framework, introduced by Moraglio et al. and enhanced by Pauckert et al., represents a contribution to converting combinatorial optimization problems into the QUBO format. The framework accepts cost and constraint functions defined in a high-level programming language such as Python. To derive the QUBO coefficients, it employs a data-driven interpolation method that samples the problem description using a special selection of binary input vectors, specifically, the all-zeros vector, one-hot encoded vectors, and two-hot encoded vectors. This process systematically determines the constant, linear, and quadratic coefficients of the expression QUBO. Subsequently, AutoQUBO v2 can construct separate QUBO matrices for the cost and constraint functions, automatically estimate a valid penalty weight based on the cost matrix, and finally combine them to produce the final QUBO matrix for the problem. (Moraglio, Georgescu, and Sadowski 2022; Pauckert et al. 2023).

However, the framework's approach to non-binary variables, particularly continuous ones, has a limitation based on direct binarization. For a problem with continuous variables, such as an Uncapacitated Facility Location Problem (UFLP) with N facilities and M customers, if each continuous assignment variable is encoded with K binary bits, the total number of variables in the QUBO model scales to $N + (N \times M \times K)$. Consequently, the size of the resulting QUBO matrix grows quadratically as $(N + (N \times M \times K))^2$. This explosive growth in dimensionality, especially with a large number of customers M or required precision K, poses a fundamental scalability bottleneck, limiting the framework's applicability for solving complex combinatorial problems that involve continuous variables.

Research Gap

While the above works have made significant progress in automating the initial step from natural language to a structured model like MILP, a critical bottleneck remains in the subsequent stage: the translation from a classical optimization model into a hardware-compatible QUBO format. This process is not only complex and error-prone, but also crucial for leveraging quantum hardware. To bridge this research gap, our work is driven by two central questions.

- How can we leverage LLMs to reliably automate the expert-driven, rule-intensive conversion of a structured MILP into a correct and efficient QUBO representation?
- How can this automated QUBO generation be integrated into a scalable, hybrid quantum-classical framework to overcome the limitations of near-term quantum devices and solve large-scale problems?

Framework Design

This section introduces the architecture of our framework, which leverages a LLM to automate the complex pipeline from a natural language problem description to a quantum-ready solution. The framework is designed to address the primary bottleneck in applying quantum computing: the manual, expert-driven conversion of optimization problems into the QUBO format.

Overall Architecture and Workflow

The architecture of our proposed framework is illustrated in Figure 1. It is an end-to-end pipeline that begins with a natural language problem description as its primary input.

The initial step of this pipeline is Stage 1: LLM-driven Problem Structuring. In this stage, the framework uses an LLM to translate the unstructured problem description into a formal mathematical model. The process involves guiding the LLM to parse the input text and identify five categories: sets, over which indices are defined; parameters, the constants of the problem; decision variables, including their types (e.g., binary, continuous); the objective function with its optimization direction; and the constraints. The LLM then synthesizes these extracted components into a structured MILP, which serves as the standardized input for subsequent transformation stages. The accurate classification of variable types at this stage is critical, as it directly informs the logic of the QUBO conversion process. From the Structured MILP, the framework supports two distinct operational workflows.

For smaller problems, a direct conversion path (Path 3a) is taken, where the MILP is fed into Stage 2: LLM-driven QUBO Transformation Engine. For large-scale problems that would exceed the capacity of current quantum hardware, the framework employs a hybrid decomposition strategy (path 3b). In this workflow, the MILP is first partitioned by a Benders' Decomposition module into a combinatorial Master Problem and a linearly structured Subproblem. The Master Problem is then sent to the Stage 2 engine for QUBO conversion (Path 4), while the Subproblem is handled by a Classical Solver.

The final QUBO matrix is dispatched to a Quantum Annealer to find an optimal solution. In the hybrid workflow, an iterative refinement process begins. The solutions from the quantum and classical solvers are used to generate Benders' cuts, which are fed back to the Benders' Decomposition module to refine the Master Problem (Path 9). This loop continues until a predefined convergence criterion is satisfied, which yields the final optimal solution to the original large-scale problem.

QUBO Transformation Engine

The second and core stage of our framework is the automated conversion of the structured MILP into a QUBO format. We address this challenge by using the LLM as an expert rule-based conversion engine through a process of structured prompt engineering.

Our methodology guides the LLM to systematically deconstruct the input MILP and resynthesize it into a structured Python class suitable for QUBO generation tools like AutoQUBO. This class explicitly separates the objective function from the constraint penalty terms. The LLM is instructed to adhere to the following key conversion principles.

- **Common Constraint Conversion:** The LLM handles standard constraints by converting them into quadratic penalty terms. Equality constraints of the form $LHS = RHS$ are directly formulated as a penalty $(LHS - RHS)^2$. For inequalities such as $LHS \leq RHS$, the process involves first introducing a binarized slack variable, s, to form an equivalent equality, $LHS + s = RHS$. This is then converted into its corresponding quadratic penalty, $(LHS + s - RHS)^2$. To conserve resources, the number of binary bits for the slack variable s is minimized based on the constraint parameters, following a "just enough" precision rule.
- **Specialized Structure Recognition:** Beyond general rules, our prompt engineering also equips the LLM to identify special constraint structures that allow for more efficient QUBO formulations. A notable example is the pairwise exclusion constraint $x_i + x_j \leq 1$, where x_i and x_j are binary variables. Instead of mechanically applying the slack variable method, the LLM is guided to recognize that this constraint is violated only when $x_i = 1$ and $x_j = 1$. This violation is captured by the more compact quadratic penalty term $P \cdot x_i x_j$, where P is a sufficiently large penalty coefficient. This specialized handling avoids the introduction of unnecessary slack variables, resulting in a more efficient final QUBO model.

To ensure that the resulting QUBO model is compatible with the target hardware, non-binary variables must be converted into a binary representation. Our framework manages this binarization process while being aware of physical hardware limitations.

An integer variable y within a range $[0, U]$ is binarized using a standard binary expansion. It is replaced by a sum of k new binary variables, $z_0, z_1, \ldots, z_{k-1}$:

$$y = \sum_{i=0}^{k-1} 2^i z_i$$

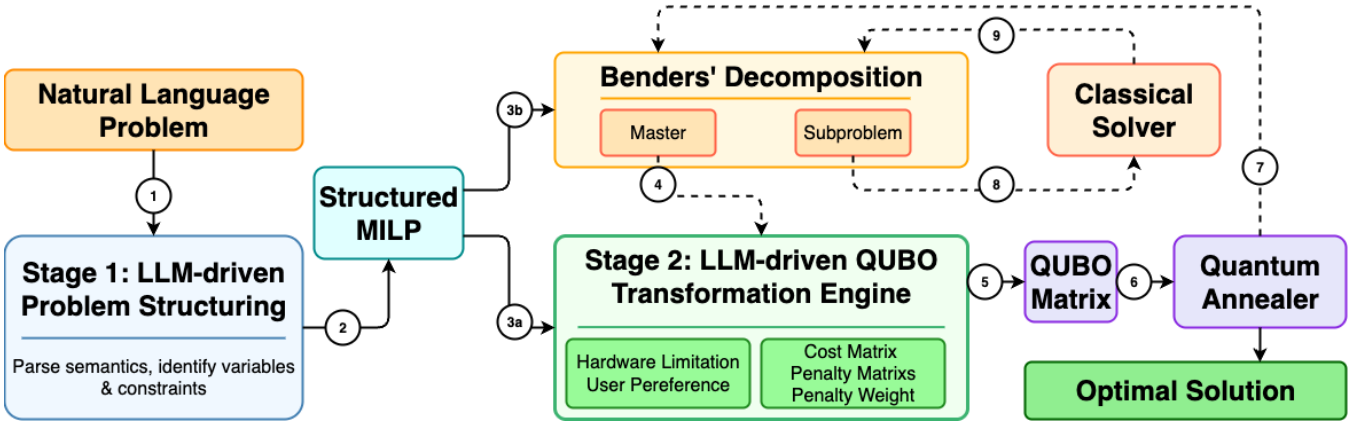

Figure 1: Framework Overview

The precision k is the minimum number of bits required to represent the upper bound U, calculated as $k = \lceil \log_2(U + 1) \rceil$. A critical aspect of our framework is its awareness of the quantum annealer capacity. For example, assume that a problem has 15 native binary variables and one integer variable and that the target quantum annealer can only accept a QUBO matrix of up to 24×24. This imposes a hard limit on the total number of variables: $15 + k \leq 24$, which implies that the binarization precision for the integer variable cannot exceed $k = 9$ bits. Our structured prompts instruct the LLM to calculate and adhere to such hardware-aware precision limits for all non-binary variables, ensuring that the final QUBO model is physically solvable.

The final QUBO matrix is the original cost matrix and a weighted sum of all penalty matrix derived from the constraints. The general form is:

$$\text{QUBO Matrix} = \text{Cost Matrix} + \sum_j P_j \cdot \text{Penalty Matrix}_j$$

where P_j is the penalty coefficient for the j-th constraint.

Although existing tools often use a single penalty weight P, this is not good enough since real-world constraints may have varying importance. Our LLM-driven approach opens the path for inferring the semantic priority of each constraint from the original problem description to assign distinct weights (P_j). For example, the LLM can assign a higher penalty to a critical physical capacity constraint than to a less rigid budget constraint. This intelligent weighting leads to a better-conditioned QUBO model that guides the solver more effectively towards a valid and optimal solution.

Application: Solving Large-Scale Problems via Hybrid Benders' Decomposition

While our LLM-driven conversion engine can process any suitable MILP, converting a large-scale problem into a single, monolithic QUBO often results in a model that is too large and complex to be solved effectively. This scalability challenge, which we demonstrate empirically in Section 4, motivates the use of a decomposition strategy. To this end, we apply our conversion methodology within a hybrid Benders' decomposition framework to tackle large-scale optimization problems.

This strategy partitions a source MILP into more manageable components. Our framework assumes the MILP can be expressed in the following general form:

$$\begin{aligned} \text{minimize} \quad & c^T x + f^T y \\ \text{subject to} \quad & Ax + By \leq b \\ & x \in \mathbb{R}^n_+, \quad y \in \{0,1\}^p \end{aligned} \tag{1}$$

Here, y represents the vector of p binary decision variables that capture the combinatorial complexity of the problem, while x is the vector of n continuous variables. The Benders' decomposition method reformulates this problem into two smaller, linked problems:

1. **The Master Problem (MP):** An integer program involving only the binary variables y. It approximates the impact of the continuous variables through a set of constraints known as Benders' cuts, which are added iteratively.
2. **The Subproblem (SP):** A linear program that solves for the continuous variables x, assuming the values of the binary variables y are fixed.

The critical integration point for our core contribution occurs in the master problem. The isolated MP, now a pure binary optimization problem, becomes the direct input for our LLM-driven conversion engine described in Stage 2. The LLM applies the rules-based methodology to transform the MP into a compact, hardware-aware QUBO.

The overall hybrid workflow proceeds iteratively. In each iteration, the MP-QUBO is solved to propose a new set of binary decisions. The SP is then solved with these decisions fixed. The solution to the SP is used to generate a new Benders' cut that is added to the MP for the next iteration, progressively refining the solution space. As established in our framework's introduction, the resulting QUBO is validated in our experiments using classical solvers. This approach confirms the correctness of the decomposition and conversion, benchmarking the structure's performance and verifying the quantum hardware compatibility of the QUBO

formed master problem for future implementation on a physical quantum annealer.

Experiment and Result Analysis

Recent work demonstrates that LLM can accurately translate natural language descriptions of optimization problems into formal MILP models, with frameworks such as LLMOPT achieving high accuracy on standard benchmarks. Given this established capability, our work begins with the assumption of an accurate pre-existing MILP model. Therefore, our experimental focus is on the automated, high-quality conversion of these MILPs into QUBO formulations.

LLM Automatic Modeling

The LLM automatic modeling tasks were performed using a Qwen3-8B model, locally deployed on a single NVIDIA A40 GPU. The primary objective of our automated conversion framework is to produce QUBO models that are not only syntactically valid but also semantically equivalent to the source MILP. We assess the quality of the conversion based on three criteria: (1) adherence to the standardized input format required by tools like AutoQUBO, which separates the cost function from constraint penalty terms; (2) the correct application of binarization strategies for non-binary decision variables and the introduction of slack variables for inequality constraints; and (3) the mathematical correctness of the penalty function formulation for each constraint.

Our analysis, summarized in Table 1, evaluates the conversion of nine classical optimization problems. The results indicate that, while many problems can be converted successfully, significant challenges arise from complex constraint structures, often leading to incorrect penalty formulations. To illustrate these findings, we analyze one correct conversion and one incorrect conversion in detail.

Correct Conversion Example: Capacitated Facility Location Problem CFLP conversion demonstrates robust and correct handling of multiple and diverse constraint types. This problem includes equality, inequality, and indicator logic constraints, all of which were transformed correctly.

- The customer service equality constraint, $\sum_i x_{ij} = 1$, was correctly penalized using the standard quadratic form $(\sum_i x_{ij} - 1)^2$.
- The indicator logic constraint, $x_{ij} \leq y_i$, which links the binary variables for customer assignment (x_{ij}) and facility opening (y_i), was efficiently converted into the compact quadratic penalty term $x_{ij}(1 - y_i)$.
- Most importantly, the capacity inequality constraint, $\sum_j d_j x_{ij} \leq C_i y_i$, was correctly transformed into an equality by introducing a binarized integer slack variable s_i. This led to the mathematically sound penalty function $(\sum_j d_j x_{ij} + s_i - C_i y_i)^2$.

The successful conversion of the multifaceted CFLP highlights that a rules-based, systematic approach can produce correct and high-quality QUBO formulations, provided that established principles for handling different constraint types are strictly followed.

Incorrect Conversion Example: Traveling Salesman Problem (TSP) The TSP conversion successfully binarized the continuous auxiliary variables u_i from the Miller-Tucker-Zemlin (MTZ) sub-tour elimination constraints. However, it fails on the third criterion regarding penalty correctness. The MTZ inequality is formulated as $u_i - u_j + n \cdot x_{ij} \leq n - 1$.

The implemented penalty term for this constraint was $\max(0, u_i - u_j + n \cdot x_{ij} - (n-1))^2$. Although this function correctly identifies violations, its expansion after substituting the binarized representation of u_i and u_j results in a polynomial of an order greater than two. This violates the quadratic nature of QUBO and creates a model that is incompatible with standard solvers. The correct approach requires the introduction of a binarized slack variable s_{ij} to form an equality $u_i - u_j + n \cdot x_{ij} + s_{ij} = n - 1$. This equality can then be correctly penalized with a standard quadratic term $(u_i - u_j + n \cdot x_{ij} + s_{ij} - (n-1))^2$.

Scalable Hybrid Computation

To evaluate the performance and scalability of our proposed HQC framework, we used the Capacitated Facility Location Problem (CFLP) as our testbed. This NP-hard problem is ideal for Benders' decomposition due to its inherent structure, ensuring a fair comparison across methodologies. All experiments were conducted on an Apple M4 Pro CPU with Gurobi 12.0.3, using problem instances from the OR-Library benchmark suite for reproducibility.

Benchmarking Methodology We compare the performance of our framework against two critical baselines, representing the state-of-the-art classical approach and the naive quantum-inspired approach, respectively.

- Method 1: Direct MILP solution (Gurobi). The complete CFLP is formulated as a standard Mixed-Integer Linear Program and solved directly using Gurobi. This serves as our baseline for both solution quality and classical performance.
- Method 2: The entire QUBO matrix solution. The entire CFLP, including binarized representations of all variables and constraints, is converted into a single large QUBO matrix using our framework. This large QUBO matrix is then solved using Gurobi's quadratic programming capabilities. This method represents the performance of a direct, nondecomposed QUBO approach.
- Method 3: Hybrid Benders Decomposition (Our Approach). The CFLP is partitioned according to our framework's logic. The master problem, which contains the binary facilities opening variables, is formulated as a QUBO. The subproblem, handling customer assignments and capacity constraints, is formulated as a Linear Program (LP). For this experiment, both the master QUBO and the LP subproblems are solved using Gurobi to provide a fair, head-to-head comparison of the algorithmic structure's efficiency against Method 1.

The Scalability Bottleneck of Monolithic QUBO Formulations Before evaluating our hybrid method, we first demonstrate the fundamental challenge it is designed to

Problem	Structure	Decision Variable	Constraint Type	Encoding Strategy	Penalty Correctness
Traveling Salesman Problem	✓	Binary `x`, Continuous `u`	Equality (Degree) & Inequality (Subtour Elimination)	Dynamic bits for `u`.	Has high order item.
Weighted Max-Satisfiability	✓	Binary `x`, `z`	Inequality (Logical Clause Association)	Dynamic bits for slack `s`.	✓
Vehicle Routing Problem	✓	Binary `x`	Equality (Flow/Visit) & Inequality (Capacity)	Dynamic bits for slack `s`.	✓
Portfolio Optimization	✓	Continuous `w`	Equality (Budget) & Inequality (Minimum Investment	Not performed.	Incorrect penalty function.
Maximum Clique	✓	Binary `x`	Inequality (Pairwise Node Exclusion)	N/A	✓
Maximum Independent Set	✓	Binary `x`	Inequality (Pairwise Node Exclusion)	N/A	✓
Logistics Network Design	✓	Binary `y`, Continuous `x`	Equality (Demand) & Inequality (Performance Metrics)	Dynamic bits for flow `x`.	✓
Knapsack Problem	✓	Binary `x`	Inequality (Single Capacity) `s`.	Dynamic bits for slack `s`.	✓
Capacitated Facility Location	✓	Binary `x`, `y`	Multiple Types (Coverage, Metrics, Capacity)	Dynamic bits for slack `s`.	✓

Table 1: Analysis of automated MILP-to-QUBO conversion correctness for nine classical optimization problems.

overcome: the poor scalability of monolithic QUBO formulations. We applied the Monolithic QUBO Solver (Method 2) to a moderately sized CFLP instance (20 facilities, 20 customers) from the Discrete Location Problems Benchmark library (Beresnev et al. 2011).

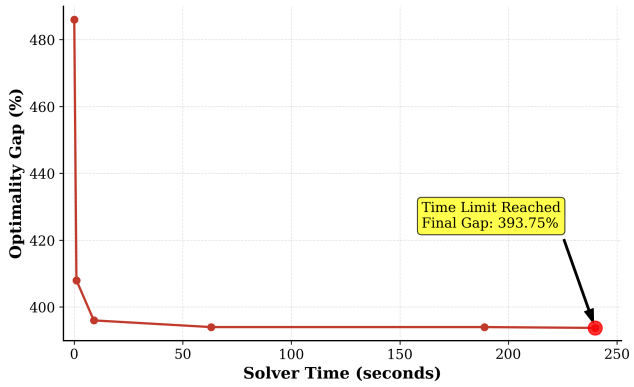

Figure 2: Convergence Failure of the Monolithic QUBO Approach on a 20x20 CFLP Instance.

Figure 2 illustrates the solver's performance in this task. The plot tracks the convergence of the optimality gap over time, showing that despite rapid initial improvements, the solver quickly stagnates. It does not reduce the gap below 393% before reaching the 240-second time limit. This result underscores the intractability of solving non-trivial optimization problems via a direct, monolithic QUBO conversion; the enormous size and complex structure of the resulting matrix create a prohibitively difficult search landscape, thereby motivating the need for intelligent decomposition.

Performance and Scalability of Hybrid Benders Decomposition We validate our framework on a suite of benchmark instances for the CFLP drawn from the well-established OR-Library, originally introduced by Ahuja et al. and Chen and Ting, and utilize the specific problem sets and their corresponding optimal solution values as benchmarked in Guastaroba and Speranza (Ahuja et al. 2004; Chen and Ting 2008; Guastaroba and Speranza 2014). In this section, we compare the performance of our Hybrid Benders Decomposition (Method 3) against the state-of-the-art Direct MILP Solver (Method 1) across these instances, with sizes scaling from 16 facilities and 50 customers up to 100 facilities and 1000 customers.

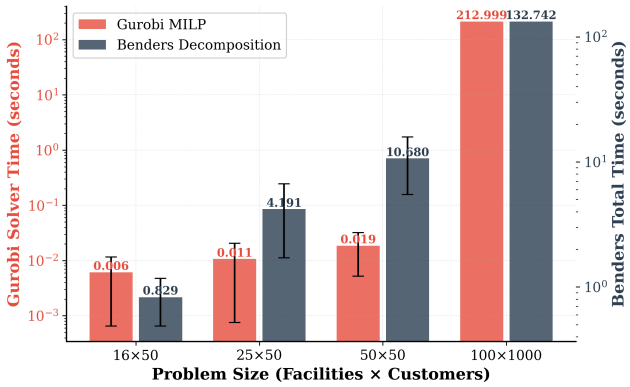

Figure 3: Scalability Comparison of the Hybrid Benders Decomposition against a Direct MILP Solver.

Figure 3 presents the core results of our scalability analysis, comparing the performance of our Hybrid Benders Decomposition (Method 3, total time, gray) versus the Gurobi

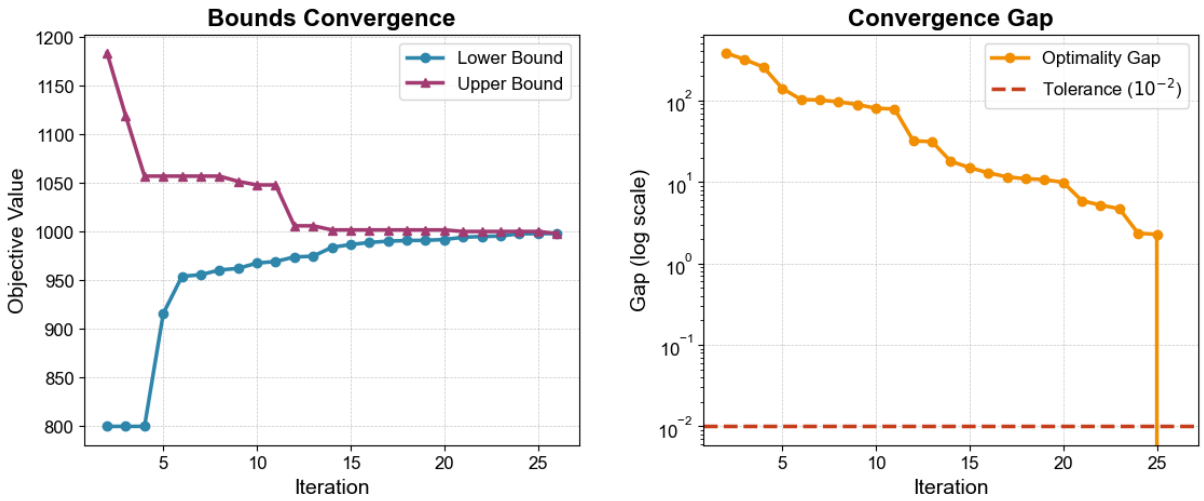

Figure 4: Overview of the convergence behavior in the CFLP Benders decomposition approach

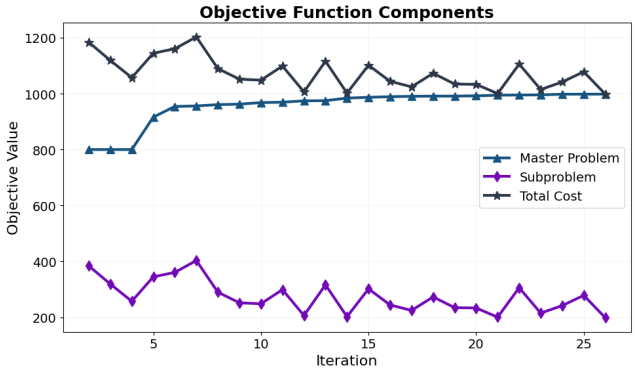

Figure 5: Evolution of Objective Function Components During Benders Decomposition.

Direct MILP Solver (Method 1, solver time, red) on CFLP instances of increasing scale. Both y-axes are on a logarithmic scale, and error bars represent one standard deviation over multiple runs. For smaller instances (e.g., 16×50 and 25×50), the overhead of the Benders iterative loop makes it slower than the highly optimized direct Gurobi solver. However, the advantage of our decomposition approach becomes strikingly evident at the largest scale (100 facilities and 1000 customers). While the direct Gurobi solver required 213.0 seconds to find and prove the optimal solution, our hybrid framework obtained a high-quality solution with a negligible optimality gap of only 0.2% in just 132.7 seconds, achieving a 38% reduction in runtime compared to the direct Gurobi solver. It is crucial to note that these compelling results were obtained by solving the QUBO master problem classically. As the master problem is designed for a quantum annealer, we anticipate further performance gains when integrating a QPU backend, positioning our framework as a potent quantum accelerator for the combinatorial core of large-scale problems.

Figures 4 and 5 provide a detailed view into the internal mechanics of the Benders' decomposition process for a representative instance. Figure 4 demonstrates the characteristic convergence pattern, where the lower bound (derived from the master problem's relaxed solutions) monotonically increases while the upper bound (from feasible solutions found by the subproblem) decreases, robustly closing the optimality gap in under 30 iterations. Figure 5 dissects the objective value to illustrate the dynamics of the algorithm. The plot shows the Master Problem objective (which forms a monotonically increasing lower bound), the Subproblem cost (representing the optimal transportation cost for a given set of open facilities), and the Total Cost (the upper bound derived from the combined feasible solution in each iteration). This visualization highlights the interplay between the combinatorial master problem and the linear subproblem as they guide the search toward convergence.

Conclusion and Future Work

In this paper, we present a novel framework that leverages an LLM to automate the end-to-end pipeline from a standard MILP model to a QUBO formulation. Our experiments demonstrate that the proposed LLM-driven conversion process is both stable and capable of producing high-quality QUBO models that are semantically equivalent to their source problems. Furthermore, by integrating this core methodology into a Benders' decomposition scheme, we have demonstrated a powerful and scalable approach for solving large-scale optimization problems. This successful integration represents a significant step towards a new paradigm of hybrid quantum-AI computation.

For future work, we plan to evolve our methodology from its current rule-based implementation to a more robust learning-based approach. The current system, which relies on structured prompt engineering, offers the significant advantage of rapid iteration and validation of conversion strategies. Having established an effective set of prin-

ciples through this method, our next step will be to use these curated data to fine-tune an LLM. By shifting from a context-based approach (providing rules in the prompt) to a learning-based one (embedding knowledge into the model's weights), we anticipate further enhancing the framework's robustness, improving its generalization capabilities across a wider range of problem classes, and potentially discovering even more efficient QUBO formulation strategies.

References

Ahmed, T.; and Choudhury, S. 2024. LM4OPT: Unveiling the Potential of Large Language Models in Formulating Mathematical Optimization Problems. *INFOR: Information Systems and Operational Research*, 62(4): 559–572.

Ahuja, R. K.; Orlin, J. B.; Pallottino, S.; Scaparra, M. P.; and Scutellà, M. G. 2004. A Multi-Exchange Heuristic for the Single-Source Capacitated Facility Location Problem. *Management Science*, 50(6): 749–760.

Ayodele, M. 2022. Penalty Weights in QUBO Formulations: Permutation Problems. In Pérez Cáceres, L.; and Verel, S., eds., *Evolutionary Computation in Combinatorial Optimization*, volume 13222, 159–174. Cham: Springer International Publishing. ISBN 978-3-031-04147-1 978-3-031-04148-8.

Beresnev, V.; Kochetov, Y.; Pashchenko, M.; Goncharov, E.; and Plyasunov, A. 2011. Discrete Location Problems Benchmark Library.

Chen, C.-H.; and Ting, C. 2008. Combining Lagrangian Heuristic and Ant Colony System to Solve the Single Source Capacitated Facility Location Problem. *Transportation Research Part E-logistics and Transportation Review*, 44(6): 1099–1122.

Chen, H.; Constante-Flores, G. E.; Mantri, K. S. I.; Kompalli, S. M.; Ahluwalia, A.; and Li, C. 2025. OptiChat: Bridging Optimization Models and Practitioners with Large Language Models. *ArXiv*, abs/2501.08406.

Glover, F. W.; Kochenberger, G. A.; Hennig, R.; and Du, Y. 2022. Quantum Bridge Analytics I: A Tutorial on Formulating and Using QUBO Models. *Ann. Oper. Res.*, 314(1): 141–183.

Guastaroba, G.; and Speranza, M. 2014. A Heuristic for BILP Problems: The Single Source Capacitated Facility Location Problem. *Eur. J. Oper. Res.*, 238(2): 438–450.

Holliday, J. 2025. Solving Real-World Optimization Problems Using Near-Term Quantum Computing with Applications in Vehicle Routing and Drone Delivery. *Graduate Theses and Dissertations*.

Huang, C.; Tang, Z.; Hu, S.; Jiang, R.; Zheng, X.; Ge, D.; Wang, B.; and Wang, Z. 2025. ORLM: A Customizable Framework in Training Large Models for Automated Optimization Modeling. *Operations Research*.

Jiang, C.; Shu, X.; Qian, H.; Lu, X.; Zhou, J.; Zhou, A.; and Yu, Y. 2025. LLMOPT: Learning to Define and Solve General Optimization Problems from Scratch. In *Proceedings of the The Thirteenth International Conference on Learning Representations*, volume abs/2410.13213. OpenReview.net.

Malviya, G.; AkashNarayanan, B.; and Seshadri, J. 2023. Logistics Network Optimization Using Quantum Annealing. In Noor, A.; Saroha, K.; Pricop, E.; Sen, A.; and Trivedi, G., eds., *Proceedings of Third Emerging Trends and Technologies on Intelligent Systems*, 401–413. Singapore: Springer Nature. ISBN 978-981-99-3963-3.

Moraglio, A.; Georgescu, S.; and Sadowski, P. 2022. AutoQubo: Data-Driven Automatic QUBO Generation. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, 2232–2239.

Morapakula, S. N.; Deshpande, S.; Yata, R.; Ubale, R.; Wad, U.; and Ikeda, K. 2025. End-to-End Portfolio Optimization with Quantum Annealing.

Mücke, S.; Gerlach, T.; and Piatkowski, N. 2023. Optimum-Preserving QUBO Parameter Compression. *Quantum Mach. Intell.*, 7: 1.

Oliveira, N. M. D.; Silva, R. M. D. A.; and Oliveira, W. R. D. 2018. QUBO Formulation for the Contact Map Overlap Problem. *International Journal of Quantum Information*, 16(08): 1840007.

Pauckert, J.; Ayodele, M.; García, M.; Georgescu, S.; and Parizy, M. 2023. AutoQUBO v2: Towards Efficient and Effective QUBO Formulations for Ising Machines. *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, 227–230.

Ramamonjison, R.; Li, H.; Yu, T. T.; He, S.; Rengan, V.; Banitalebi-Dehkordi, A.; Zhou, Z.; and Zhang, Y. 2022. Augmenting Operations Research with Auto-Formulation of Optimization Models from Problem Descriptions.

Ramamonjison, R.; Yu, T. T. L.; Li, R.; Li, H.; Carenini, G.; Ghaddar, B.; He, S.; Mostajabdaveh, M.; Banitalebi{-}Dehkordi, A.; Zhou, Z.; and Zhang, Y. 2021. NL4Opt Competition: Formulating Optimization Problems Based on Their Natural Language Descriptions. In *Proceedings of the NeurIPS 2022 Competition Track*, volume 220 of *Proceedings of Machine Learning Research*, 189–203. PMLR.

Volpe, D.; Quetschlich, N.; Graziano, M.; Turvani, G.; and Wille, R. 2024. Towards an Automatic Framework for Solving Optimization Problems with Quantum Computers. In *2024 IEEE International Conference on Quantum Software (QSW)*, 46–57.

Zaman, M.; Tanahashi, K.; and Tanaka, S. 2022. PyQUBO: Python Library for Mapping Combinatorial Optimization Problems to QUBO Form. *IEEE Trans. Computers*, 71: 838–850.

Zhao, Z.; Fan, L.; and Han, Z. 2022. Hybrid Quantum Benders' Decomposition For Mixed-integer Linear Programming. *2022 IEEE Wireless Communications and Networking Conference (WCNC)*, 2536–2540.

Zhao, Z.; Li, M.; Fan, L.; and Han, Z. 2025. HQC-Bend: A Python Package of Hybrid Quantum-Classical Multi-cuts Benders' Decomposition Algorithm. *2025 International Conference on Quantum Communications, Networking, and Computing (QCNC)*, 591–597.